Оглавление

1 Общая архитектура системы **4**

2 Конфигурационные файлы **5**

2.1 Полномочия пользователей . . . 5
2.2 Ограничение доступа по IP-адресам . . . 7
2.2.1 Ограничения IP-адресов в формате XML . . . 8
2.2.2 Элемент `access` . . . 9
2.2.3 Элемент `ip` . . . 9
2.2.4 Пример использования элемента `access` . . . 10
2.2.5 Ограничения IP-адресов в текстовом формате . . . 10
2.3 Локализация системы . . . 11
2.3.1 Включение поддержки языкового окружения . . . 11
2.3.2 Установка каталога сообщений . . . 12
2.3.3 Поддержка кодировок символов . . . 12
2.4 Общий конфигурационный файл `ejudge.xml` . . . 13
2.4.1 Общая структура . . . 14
2.4.2 Тег `config` . . . 14
2.4.3 Тег `cap` . . . 15
2.4.4 Тег `caps` . . . 15
2.4.5 Тег `contests_dir` . . . 16
2.4.6 Тег `email_program` . . . 16
2.4.7 Тег `l10n_dir` . . . 17
2.4.8 Тег `map` . . . 17
2.4.9 Тег `register_email` . . . 18
2.4.10 Тег `register_url` . . . 18
2.4.11 Тег `serve_path` . . . 19
2.4.12 Тег `run_path` . . . 19
2.4.13 Тег `socket_path` . . . 19
2.4.14 Тег `user_map` . . . 21
2.4.15 Тег `userdb_file` . . . 21
2.5 Конфигурационный файл `register.xml` . . . 22
2.5.1 Тег `register_config` . . . 22
2.5.2 Тег `access` . . . 24
2.5.3 Тег `ip` . . . 24
2.5.4 Тег `l10n_dir` . . . 25
2.5.5 Тег `contests_dir` . . . 25
2.5.6 Тег `socket_path` . . . 26
2.6 Конфигурационный файл `users.xml` . . . 27

2.6.1 Тег users_config . . . 27
2.6.2 Тег access . . . 29
2.6.3 Тег ip . . . 29
2.6.4 Тег l10n_dir . . . 30
2.6.5 Тег contests_dir . . . 30
2.6.6 Тег socket_path . . . 31
2.7 Конфигурационный файл master.cfg . . . 31
2.7.1 Параметр allow_deny . . . 32
2.7.2 Параметр allow_from . . . 32
2.7.3 Параметр deny_from . . . 32
2.7.4 Параметр charset . . . 32
2.7.5 Параметр enable_l10n . . . 33
2.7.6 Параметр l10n_dir . . . 33
2.7.7 Параметр socket_path . . . 33
2.7.8 Параметр contests_dir . . . 34
2.8 Конфигурационный файл judge.cfg . . . 34
2.8.1 Параметр allow_deny . . . 35
2.8.2 Параметр allow_from . . . 35
2.8.3 Параметр deny_from . . . 35
2.8.4 Параметр charset . . . 35
2.8.5 Параметр enable_l10n . . . 36
2.8.6 Параметр l10n_dir . . . 36
2.8.7 Параметр socket_path . . . 36
2.8.8 Параметр contests_dir . . . 37
2.9 Конфигурационный файл team.cfg . . . 37
2.9.1 Параметр allow_deny . . . 38
2.9.2 Параметр allow_from . . . 38
2.9.3 Параметр deny_from . . . 38
2.9.4 Параметр charset . . . 38
2.9.5 Параметр enable_l10n . . . 39
2.9.6 Параметр l10n_dir . . . 39
2.9.7 Параметр socket_path . . . 39
2.9.8 Параметр contests_dir . . . 40
2.9.9 Параметр show_generation_time . . . 40
2.10 Конфигурационный файл турниров contest.xml . . . 41
2.10.1 Элемент contest . . . 45
2.10.2 Элемент name . . . 48
2.10.3 Элемент name_en . . . 48
2.10.4 Элемент register_access . . . 48
2.10.5 Элемент users_access . . . 49
2.10.6 Элемент master_access . . . 49
2.10.7 Элемент judge_access . . . 49
2.10.8 Элемент team_access . . . 50
2.10.9 Элемент ip . . . 50
2.10.10 Элемент field . . . 50
2.10.11 Элементы contestants, reserves, coaches, advisors, guests 52
2.10.12 Элемент users_header_file . . . 54
2.10.13 Элемент users_footer_file . . . 54

2.10.14 Элемент register_header_file . . . 55
2.10.15 Элемент register_footer_file . . . 55
2.10.16 Элемент team_header_file . . . 56
2.10.17 Элемент team_footer_file . . . 56
2.10.18 Элемент users_head_style . . . 57
2.10.19 Элемент users_par_style . . . 58
2.10.20 Элемент users_table_style . . . 59
2.10.21 Элемент users_verb_style . . . 60
2.10.22 Элемент register_head_style . . . 61
2.10.23 Элемент register_par_style . . . 62
2.10.24 Элемент register_table_style . . . 63
2.10.25 Элемент team_head_style . . . 64
2.10.26 Элемент team_par_style . . . 65
2.10.27 Элемент register_email . . . 66
2.10.28 Элемент register_url . . . 66
2.10.29 Элемент team_url . . . 67
2.10.30 Элемент registration_deadline . . . 67
2.11 Конфигурационный файл сервера турнира serve.cfg . . . 69
2.11.1 Общая структура . . . 69
2.11.2 Форматные подстановки . . . 70
2.11.3 Список глобальных параметров . . . 72
2.11.4 Глобальные параметры . . . 78
2.11.5 Параметры задачи . . . 147
2.11.6 Параметры языка . . . 176
2.11.7 Параметры тестировщика . . . 183
2.12 Файл описания теста test.inf . . . 205
2.12.1 Использование . . . 205
2.12.2 Общая структура . . . 205
2.12.3 Конфигурационные переменные . . . 207

Глава 1

Общая архитектура системы

Система ejudge включает в себя несколько компонент, обменивающихся информацией через общие каталоги или через UNIX-сокеты. Компоненты первого типа могут быть распределёнными на несколько машин, компоненты второго типа должны выполняться на одной (серверной) машине.

В центре системы находятся две компоненты — userlist-server и serve. Компонента userlist-server поддерживает базу данных зарегистрированных пользователей, выполняя запросы как от пользователей (через cgi-программы users, register), так и от администратора базы (через программу edit-userlist). Все запросы принимаются только через локальный UNIX-сокет, поэтому клиенты могут работать только на том же самом компьютере.

Компонента serve выполняет функции сервера турнира, то есть принимает от пользователей информационные запросы и решения отосланные на проверку, выдаёт полученные решения на компиляцию и проверку, обновляет текущие результаты и т. д. Запросы от пользователей и администратора турнира принимаются по локальному UNIX-сокету. Как правило, эти запросы формируются cgi-программами master и team. Обмен с компонентами compile и run ведётся через общие каталоги, что позволяет размещать их на нескольких компьютерах, разделяющих общую сетевую файловую систему.

Программа compile компилирует поступивший ей файл с исходным текстом программы. Скомпилированная программа при успешной компиляции или список ошибок при неуспешной компиляции возвращаются серверу турнира также через общие каталоги.

Программа run запускает поступившую ей от сервера скомпилированную программу на заданном наборе тестов и в зависимости от режима проверки формирует ответ, который передаётся обратно серверу вместе с журналом проверки.

Глава 2

Конфигурационные файлы

В данном разделе описываются конфигурационные файлы системы.

2.1 Полномочия пользователей

Система ejudge поддерживает тонкую настройку прав выполнения всех привилегированных операций для пользователей системы. К привилегированным операциям относятся операции управления турниром (запуск, останов, просмотр и изменения результатов проверки программы и пр.) и операции управления базой данных пользователей. Настройка осуществляется с помощью указаний полномочий пользователя для выполнения операций над базой пользователей и полномочий пользователя для выполнения операций над каждым из созданных в системе турниров.

Пользователи, для которых указаны полномочия выполнения операций над базой пользователей, считаются привилегированными. При этом может быть указано пустое множество полномочий. Пользователи, для которых указаны полномочия выполнения операций над каким-либо турниром, являются привилегированными только для данного турнира, но не для системы в целом. Различие между общесистемно привилегированным пользователем (даже с пустым множеством привилегий) и всеми прочими пользователями состоит в том, что для выполнения операций над привилегированными пользователями требуются специальные привилегии.

Все привилегии системы ejudge перечислены ниже.

Название	Бит	Где	Описание
MASTER_LOGIN	0	C	Пользователь может использовать CGI-программу master.
JUDGE_LOGIN	1	C	Пользователь может использовать CGI-программу judge.
OBSERVER_LOGIN	2	C	Пользователь может использовать CGI-программу observer.
MAP_CONTEST	3	C	Пользователь может запускать сервер турнира программу serve.
LIST_CONTEST_USERS	4	C	Пользователь может просматривать список пользователей, зарегистрировавшихся на турнир.

Продолжение на следующей странице

Название	Бит	Где	Описание
LIST_ALL_USERS	5	U	Пользователь может просматривать список всех пользователей в базе пользователей.
CREATE_USER	6	U	Пользователь может создавать новых пользователей.
GET_USER	7	CU	Пользователь может получать информацию о другом непривилегированном пользователе.
EDIT_USER	8	CU	Пользователь может редактировать информацию о другом непривилегированном пользователе.
DELETE_USER	9	U	Пользователь может удалять другого непривилегированного пользователя из базы пользователей.
PRIV_EDIT_USER	10	CU	Пользователь может редактировать другого информацию о другом привилегированном пользователе.
PRIV_DELETE_USER	11	U	Пользователь может удалять другого привилегированного пользователя из базы пользователей.
GENERATE_TEAM_PASSWORDS	12	C	Пользователь может генерировать пароли для CGI-программы team для всех других непривилегированных пользователей, зарегистрировавшихся на данный турнир.
CREATE_REG	13	C	Пользователь может регистрировать других непривилегированных пользователей для участия в турнире.
EDIT_REG	14	C	Пользователь может редактировать статус регистрации других пользователей в турнире.
DELETE_REG	15	C	Пользователь может удалять регистрацию других непривилегированных пользователей в турнире.
PRIV_CREATE_REG	16	C	Пользователь может регистрировать других привилегированных пользователей для участия в турнире.
PRIV_DELETE_REG	17	C	Пользователь может удалять регистрацию других привилегированных пользователей в турнире.
DUMP_USERS	18	CU	Пользователь может получать базу пользователей в CSV-формате.
DUMP_RUNS	19	C	Пользователь может получать базу попыток сдачи в CSV-формате.
DUMP_STANDINGS	20	C	Пользователь может получать результаты турнира в CSV-формате.
VIEW_STANDINGS	21	C	Пользователь может просматривать судейскую таблицу результатов, которая никогда не замораживается.
VIEW_SOURCE	22	C	Пользователь может просматривать исходный код попыток сдачи программ участников.
VIEW_REPORT	23	C	Пользователь может просматривать отчёт о тестировании программ участников.
VIEW_CLAR	24	C	Пользователь может просматривать вопросы к судьям от участников и ответы судей.
EDIT_RUN	25	C	Пользователь может произвольным образом изменять информацию о попытке участника.

Продолжение на следующей странице

Название	Бит	Где	Описание
REJUDGE_RUN	26	C	Пользователь может отправить попытку участника на перетестирование.
NEW_MESSAGE	27	C	Пользователь может послать сообщение произвольному участнику.
REPLY_MESSAGE	28	C	Пользователь может ответить на сообщение от участника.
CONTROL_CONTEST	29	C	Пользователь может управлять турниром (стартовать, останавливать и т. д.).

Полномочия пользователей для базы данных пользователей задаются в общем конфигурационном файле ejudge.xml с помощью элемента cap. В этом элементе через запятую перечисляются полномочия указанного пользователя. Пробельные символы игнорируются. Например:

```
<cap login="vasya">
  LIST_ALL_USERS,
  GET_USER,
</cap>
```

Текст в элементе cap может быть пустым, тогда множество полномочий пользователя пусто, но он будет считаться привилегированным пользователем. Например:

```
<cap login="petya"/>
```

Полномочия пользователя по отношению к турниру задаются в конфигурационном файле турнира contest.xml с помощью элемента cap. В элементе через запятую перечисляются полномочия указанного пользователя. Пустое множество полномочий для турнира допускается, но не имеет смысла.

2.2 Ограничение доступа по IP-адресам

Система ejudge включает в себя несколько программ: register, users, master, judge, team, — запускаемых как CGI-программы. Каждая из этих программ позволяет ограничивать диапазон IP-адресов клиента, которым разрешается использование CGI-программы. В текущей версии системы ограничение диапазонов возможно только с помощью указания IPv4-адресов клиентов. Ни DNS-имена, ни IP протокол версии 6 в текущей версии не поддерживаются.

Кроме того, каждый турнир может накладывать дополнительные ограничения на допустимые IP-адреса для всех CGI-программ. Эти ограничения указываются в конфигурационном файле турнира contest.xml.

Спецификация ограничения адреса может либо задавать конкретный IP-адрес, либо задавать адрес сети классов A, B или C. Задание подсетей в сетях этих классов в настоящее время не поддерживается. Допустимые спецификации ограничения IP-адреса перечислены в таблице 2.2.

Синтаксис задания IP-ограничений отличается для CGI-программ, использующих конфигурационные файлы в формате XML (register, users), и для CGI-программ, использующих традиционные текстовые файлы (master, judge, team). Далее рассматриваются оба варианта. В конфигурационном файле турнира contest.xml ограничения задаются в формате XML.

A.B.C.D	Здесь A, B, C и D — десятичные числа от 0 до 255. В этом случае задаётся один IP-адрес.
A.B.C.	A, B и C — десятичные числа от 0 до 255. В этом случае задаётся IP-адрес сети класса C. Обратите внимание на заключительную точку в записи шаблона.
A.B.	A и B — десятичные числа от 0 до 255. В этом случае задаётся IP-адрес сети класса B. Обратите внимание на заключительную точку в записи шаблона.
A.	A — десятичное число от 0 до 255. В этом случае задаётся IP-адрес сети класса A. Обратите внимание на заключительную точку в записи шаблона.

Таблица 2.2: Допустимые спецификации ограничения IP-адреса

2.2.1 Ограничения IP-адресов в формате XML

Спецификация ограничений на IP-адрес представляет собой список спецификаций IP-адресов, для каждого из которых указано, допустимо ли использование CGI-программы клиентом с IP-адресом, удовлетворяющим спецификации, или нет. Элементы списка задаются элементом ip XML-файла. Описание этого элемента дано ниже.

Для конфигурационных файлов программ (register и users), элементы списка находятся в элементе access (см. описание элемента access конфигурационного файла register.xml и описание элемента access конфигурационного файла users.xml). Для конфигурационного файла турнира contest.xml используются элементы register_access, users_access, master_access, judge_access, team_access, задающие дополнительные ограничения на допустимые IP-адреса для программ register, users, master, judge и team соответственно. Далее даётся описание XML-элемента access, справедливое для всех вышеперечисленных элементов.

При работе CGI-программы список ограничений просматривается последовательно от первого элемента к последнему. Как только будет найдена первая спецификация, которой удовлетворяет IP-адрес клиента, дальнейший просмотр прекращается и выполняется действие, указанное в этой спецификации. Если IP-адрес клиента не удовлетворяет ни одной спецификации, выполняется действие по умолчанию, заданное в элементе верхнего уровня.

2.2.2 Элемент `access`

Имя элемента: **`access`**
Может содержать: `ip`
Атрибуты: `default`
Тип содержимого: игнорируется
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент является элементом верхнего уровня списка ограничений IP-адресов для CGI-программ.

Атрибут `default`

Имя атрибута: **`default`**
Содержится в: `access`
Тип значения: *allow* или *deny*
Значение по умолчанию: *deny*
Может отсутствовать: *да*

Описание. Данный атрибут задаёт права доступа в случае, если IP-адрес клиента не перечислен в списке IP-адресов. Если атрибут установлен в *allow*, такие IP-адреса считаются допустимыми, а если атрибут установлен в *deny*, доступ с таких IP-адресов запрещается.

2.2.3 Элемент `ip`

Имя элемента: **`ip`**
Содержится в: `access`
Может содержать: *нет*
Атрибуты: `allow`
Тип содержимого: шаблон IP-адреса (см. табл. 2.2)
Может отсутствовать: *да*
Может повторяться: *да*

Описание. Данный элемент определяет права доступа для клиентов с заданным шаблоном IP-адреса. Если атрибут `allow` установлен в *true*, таким клиентам разрешается работа с соответствующей CGI-программой. В противном случае доступ запрещается.

Атрибут `allow`

Имя атрибута: **`allow`**
Содержится в: `access`
Тип значения: *boolean*
Значение по умолчанию: *false*
Может отсутствовать: *да*

Описание. Данный атрибут задаёт права доступа в случае, если IP-адрес клиента удовлетворяет спецификации IP-адреса в данном элементе. Если значение атрибута равно *true*, такой IP-адрес считаются допустимым, а если значение атрибут установлено в *false*, доступ с этого IP-адреса запрещается.

2.2.4 Пример использования элемента `access`

```
<access default="deny">
  <ip allow="yes">127.</ip>
  <ip allow="no">192.168.0.14</ip>
  <ip allow="yes">192.168.0.</ip>
</access>
```

Данная спецификация ограничений IP-адресов задаёт, что клиентам, находящимся в сети класса A с адресом `127.` (то есть, соединениям, устанавливаемым с того же компьютера, на котором работает WEB-сервер и CGI-программы) доступ открыт. Всем клиентам, находящимся в сети класса C с адресом `192.168.0` (то есть всем клиентам, имеющим IP-адрес в диапазоне `192.168.0.1`—`192.168.0.254`), за исключением клиента с IP-адресом `192.168.0.14` доступ также открыт. Всем прочим клиентам доступ закрыт.

2.2.5 Ограничения IP-адресов в текстовом формате

Текстовый формат конфигурационного файла используется в текущей версии системы для CGI-программ master, judge и team. Для задания ограничений на допустимые IP-адреса используются параметры конфигурационного файла `allow_from`, `deny_from` и `allow_deny`.

Значениями параметров `allow_from` и `deny_from` является текстовая строка, в которой перечислены спецификации IP-адресов (см. табл. 2.2), разделённые пробелами. Параметр `allow_deny` принимает булевское значение. Значение *false* устанавливается по умолчанию (то есть если этот параметр не задан), либо может быть указано явно константой `0`. Значение *true* устанавливается, либо если в конфигурационном файле присутствует только имя параметра без указания значения, либо при явном указании значения `1`.

Параметр `allow_deny` определяет порядок просмотра IP-адресов, заданных в параметрах `allow_from` и `deny_from`.

- Если параметр `allow_deny` установлен в *false*, включается режим «недоступный по умолчанию». В этом случае IP-адрес клиента сначала сопоставляется с IP-спецификациями, перечисленными в параметре `deny_from`. Если какая-либо из IP-спецификаций сопоставляется успешно, доступ этому клиенту запрещается. Если IP-адрес не удовлетворяет ни одной из спецификаций параметра `deny_from`, IP-адрес клиента сопоставляется с IP-спецификациями, перечисленными в параметре `allow_from`. Если какая-либо из IP-спецификаций сопоставляется успешно, клиенту доступ предоставляется. Если IP-адрес не удовлетворяет ни одной из спецификаций параметра `allow_from`, доступ этому клиенту запрещается.

- Если параметр `allow_deny` установлен в *true*, включается режим «доступный по умолчанию». В этом случае IP-адрес клиента сначала сопоставляется с IP-спецификациями, перечисленными в параметре `allow_from`. Если какая-либо из IP-спецификаций сопоставляется успешно, доступ этому клиенту предоставляется. Если IP-адрес не удовлетворяет ни одной из спецификаций параметра `allow_from`, IP-адрес клиента сопоставляется с IP-спецификациями, перечисленными в параметре `deny_from`. Если какая-либо из IP-спецификаций сопоставляется успешно, доступ клиенту запрещается. Если IP-адрес не удовлетворяет ни одной из спецификаций параметра `deny_from`, доступ этому клиенту предоставляется.

В примере

```
allow_deny
```

доступ открывается клиентам со всех адресов.

В примере

```
allow_deny = 1
allow_from = "192.168.0.1 192.168.1."
deny_from = "192.168."
```

доступ открывается клиентом со всех адресов, IP-адреса которых не начинаются с `192.168`. Из клиентов с IP-адресами, начинающимися с `192.168` допускаются лишь клиенты с IP-адресом, начинающимся на `192.168.1` и клиенты с IP-адресом `192.168.0.1`.

В примере

```
allow_deny = 0
allow_from = "127. 192.168.0.1 192.168.1."
```

доступ закрывается всем, кроме клиентов с IP-адресом, начинающихся с `127` (то есть для локально устанавливаемых соединений), клиентом с IP-адресом, начинающимся с `192.168.1` и клиентов с IP-адресом `192.168.0.1`.

2.3 Локализация системы

Система ejudge поддерживает настройку языка пользовательского интерфейса, хотя текущая версия системы обладает рядом недостатков.

Переключения языка пользовательского интерфейса выполняется через стандартный механизм трансляции сообщений `gettext`. В настоящее время система поддерживает только два языковых окружения — `C` (окружение по умолчанию для программ на Си) и `ru_RU.KOI8-R` (русский язык в кодировке koi-r), которые имеют идентификаторы 0 и 1 соответственно. Для того, чтобы добавить новое языковое окружение, необходимо модифицировать исходный файл `l10n.c`. Переведённые тексты сообщений находятся в файлах `ejudge.`*LOCALE*`.po`, где *LOCALE* — имя языкового окружения.

2.3.1 Включение поддержки языкового окружения

Чтобы включить поддержку языкового окружения, необходимо установить конфигурационный параметр в файлах настройки утилит. Для конфигурационных файлов в формате XML (`ejudge.xml` `register.xml`, `users.xml`) это атрибут `l10n` элемента верхнего уровня конфигурационного файла. Например, для конфигурационного файла `ejudge.xml` установка атрибута локализации сообщений может выглядеть следующим образом:

```
<?xml version="1.0" encoding="koi8-r"?>
<config l10n="yes">
  <!-- прочие элементы -->
</config>
```

Для конфигурационных файлов в текстовом формате (`serve.cfg`, `master.cfg`, `judge.cfg`, `team.cfg`) это параметр `enable_l10n`, который необходимо установить. Для любого конфигурационного установка атрибута локализации может выглядеть следующим образом:

```
enable_l10n
```

2.3.2 Установка каталога сообщений

Чтобы система ejudge смогла использовать переводы сообщений, в конфигурационных файлах утилит необходимо указать каталог сообщений (message catalog), в котором находятся файлы с переведёнными сообщениями. По умолчанию после выполнения команды `make install` каталоги сообщений помещаются в каталог, определяемой переменной `INST_LOCALE_PATH` в `makefile`. Именно этот каталог должен быть указан в конфигурационных файлах. Для конфигурационных файлов в формате XML (`ejudge.xml register.xml`, `users.xml`) каталог сообщений устанавливается с помощью элемента `l10n_dir`, например, следующим образом:

```
<l10n_dir>/usr/share/locale</l10n_dir>
```

Для конфигурационных файлов в текстовом формате (`serve.cfg`, `master.cfg`, `judge.cfg`, `team.cfg`) каталог сообщений устанавливается с помощью параметра `l10n_dir`, например, следующим образом:

```
l10n_dir = "/usr/share/locale"
```

2.3.3 Поддержка кодировок символов

Поддержка кодировок для ввода/вывода реализована в текущей версии системы недостаточным образом. Внутренней кодировкой для системы ejudge в настоящее время является кодировка koi8-r. Все текстовые строки представляются в памяти именно в этой кодировке. Это значит, что текстовые строки в других кодировках окажутся искажены. Корректная реализация системы должны хранить строки в кодировке utf-8 или utf-16. Кроме того, пользователь должен иметь возможность задавать желательную кодировку символов.

Кодировка конфигурационных файлов

Кодировка конфигурационных файлов в формате XML (`ejudge.xml register.xml`, `users.xml`, `contest.xml`, `userlist.xml`) задаётся непосредственно в заголовке соответствующего файла, например,

```
<?xml version="1.0" encoding="koi8-r"?>
```

Текущая версия системы поддерживает следующие кодировки XML-файлов: iso8859-1, utf-8, utf-16, koi8-r, cp1251, cp866, iso8859-5. Однако при чтении файла текст в любой кодировке конвертируется в текст в кодировке koi8-r.

Кодировка конфигурационных файлов в текстовом формате (`serve.cfg`, `master.cfg`, `judge.cfg`, `team.cfg`) в настоящее время не может быть задана явно, и всегда подразумевается кодировка koi8-r.

Кодировка генерируемых файлов

При генерации html-страниц используется кодировка, заданная в конфигурационном файле соответствующей утилиты. Если параметр кодировки не задан, используется кодировка iso8859-1. В текущей версии системы ejudge параметр кодировки вывода используется только для корректной генерации http-заголовка. Не делается попытка перекодировки символов, если кодировка какой-либо части генерируемой html-страницы (например, имён пользователей) не соответствует указанной кодировке. Это практически наверняка приведёт к некорректному отображению страницы, если указана кодировка вывода, отличная от iso8859-1 или koi8-r.

Исключением в этой ситуации является таблица текущих результатов турнира. Кодировка этой html-страницы может быть задана явно с помощью параметра `standings_charset` конфигурационного файла `serve.cfg`. Если эта кодировка отличается от кодировки koi8-r, будет выполнено необходимое преобразование.

Для конфигурационных файлов утилит в формате XML (`register.xml`, `users.xml`) кодировка генерируемой страницы указывается с помощью атрибута `charset` элемента верхнего уровня. Например, в конфигурационном файле `register.xml` кодировка может быть задана следующим образом:

```
<?xml version="1.0" encoding="koi8-r"?>
<register_config
  charset = "KOI8-R"
  <!-- прочие атрибуты -->
>
  <!-- прочие элементы -->
</register_config>
```

Для конфигурационных файлов утилит в текстовом формате (`serve.cfg`, `master.cfg`, `judge.cfg`, `team.cfg`) кодировка генерируемых html-страниц указывается с помощью параметра `charset`, например, следующим образом: verbatim

```
charset = "koi8-r"
```

2.4 Общий конфигурационный файл `ejudge.xml`

Данный конфигурационный файл, который далее в тексте этого документа будет называться `ejudge.xml` содержит настройки утилит командной строки, как непосредственно работающих с базой данных, так и являющихся клиентами сервера пользователей. Файл используется утилитами clean-users, edit-userlist, super-serve, userlist-server.

2.4.1 Общая структура

Конфигурационный файл должен быть корректным XML-файлом. Он должен состоять из единственного элемента первого уровня `config`. Иерархия элементов приведена на схеме ниже.

```
config
    caps
        cap
    contests_dir
    email_program
    l10n_dir
    register_email
    register_url
    serve_path
    run_path
    socket_path
    userdb_file
    user_map
        map
```

Описание каждого элемента дано ниже.

2.4.2 Тег `config`

Имя элемента: **`config`**
Содержится в: *нет*
Может содержать: `caps`, `contests_dir`, `email_program`, `l10n_dir`, `register_email`, `register_url`, `serve_path`, `run_path`, `socket_path`, `userdb_file`, `user_map`
Атрибуты: `l10n`
Тип содержимого: игнорируется
Может отсутствовать: *нет*
Может повторяться: *нет*
Используется в: clean-users, edit-userlist, super-serve, userlist-server
Описание. Тег верхнего уровня конфигурационного файла.

2.4.3 Тег `cap`

Имя элемента: **`cap`**
Содержится в: `caps`
Может содержать: *нет*
Атрибуты: `login`
Тип содержимого: список битов полномочий
Может отсутствовать: *да*
Может повторяться: *да*
Используется в: userlist-server

Описание. Данный элемент позволяет задавать полномочия на выполнение операций над базой данных пользователей. Описание полномочий, поддерживаемых системой ejudge, приведено в разделе 2.1.

Атрибут `login` задаёт регистрационное имя пользователя, для которого устанавливаются полномочия.

Пример.

```
<cap login="dima">MAP_CONTEST</cap>
```

2.4.4 Тег `caps`

Имя элемента: **`caps`**
Содержится в: `config`
Может содержать: `cap`
Атрибуты: *нет*
Тип содержимого: игнорируется
Может отсутствовать: *да*
Может повторяться: *да*
Используется в: userlist-server

Описание. Данный элемент является элементом верхнего уровня списка списков полномочий пользователей по отношению к базе данных пользователей. Элемент внутри себя может содержать только элементы `cap`. Если элемент `caps` встречается в конфигурационном файле несколько раз, списки списков полномочий сливаются друг с другом.

Пример.

```
<caps>
  <cap login="a"></cap>
  <cap login="b"></cap>
</caps>
```

2.4.5 Тег `contests_dir`

Имя элемента: **`contests_dir`**
Содержится в: `config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к каталогу
Может отсутствовать: *нет*
Может повторяться: *нет*
Используется в: clean-users, edit-userlist, super-serve, userlist-server

Описание. Данный элемент задаёт путь к каталогу, в котором хранятся описания турниров `contest.xml`. Каждое описание турнира находится в отдельном файле с именем, равным шестизначному номеру турнира, например, `000000.xml`.

Пример.

```
<contests_dir>/var/ejudge/contests</contests_dir>
```

2.4.6 Тег `email_program`

Имя элемента: **`email_program`**
Содержится в: `config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к программе
Может отсутствовать: *нет*
Может повторяться: *нет*
Используется в: userlist-server

Описание. Данный элемент задаёт путь программе отсылки электронной почты. Эта программа запускается без аргументов командной строки, и на её стандартный поток ввода подаются служебные поля письма (`From`, `To` и т. д.), а затем текст письма. Почтовое сообщение отсылается по адресу, указанному в регистрационной анкете, при регистрации нового пользователя и содержит его регистрационное имя (login) и пароль.

Пример.

```
<email_program>/var/qmail/bin/qmail-inject</email_program>
```

2.4.7 Тег `l10n_dir`

Имя элемента: `l10n_dir`
Содержится в: `config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к каталогу
Может отсутствовать: *нет*
Может повторяться: *нет*
Используется в: userlist-server

Описание. Данный элемент задаёт путь к каталогу, в котором хранятся локализованные сообщения. Для локализации сообщений используется GNU gettext. См. раздел ??? для информации о локализации системы.

Пример.

```
<l10n_dir>/usr/share/ejudge/locale</l10n_dir>
```

2.4.8 Тег `map`

Имя элемента: `map`
Содержится в: `user_map`
Может содержать: *нет*
Атрибуты: `system_user` `local_user`
Тип содержимого: игнорируется
Может отсутствовать: *да*
Может повторяться: *да*
Используется в: userlist-server

Описание. Данный элемент задаёт отображение регистрационных имён UNIX в регистрационные имена системы ejudge. Атрибут `system_user` задаёт регистрационное имя UNIX. Пользователь с таким регистрационным именем должен существовать в системе. Атрибут `local_user` задаёт регистрационное имя системы ejudge.

Пример.

```
<map system_user="aaa" local_user="tolik"/>
```

2.4.9 Тег `register_email`

Имя элемента: **`register_email`**
Содержится в: `config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: адрес электронной почты
Может отсутствовать: *да*
Может повторяться: *нет*
Используется в: userlist-server

Описание. Данный элемент задаёт электронный адрес, от имени которого рассылаются уведомления о регистрации новых пользователей. Если этот элемент неопределён, используется адрес по умолчанию `team@contest.cmc.msu.ru`. Значение, устанавливаемое этим элементом может быть переопределено элементом `register_email` конфигурационного файла турнира `contest.xml`.

Пример.

```
<register_email>ejudge@olympiads.ru</register_email>
```

2.4.10 Тег `register_url`

Имя элемента: **`register_url`**
Содержится в: `config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: URL
Может отсутствовать: *да*
Может повторяться: *нет*
Используется в: userlist-server

Описание. Данный элемент задаёт URL CGI программы register, которая отвечает за регистрацию новых пользователей, изменение пользователями своих персональных данных и регистрацию пользователей на турниры. Этот URL добавляется в текст уведомления о регистрации. Если этот элемент неопределён, используется URL по умолчанию `http://contest.cmc.msu.ru/cgi-bin/register`. Значение, устанавливаемое этим элементом может быть переопределено элементом `register_url` конфигурационного файла турнира `contest.xml`.

Пример.

```
<register_url>http://www.olympiads.ru/cgi-bin/register</register_url>
```

2.4.11 Тег `serve_path`

Имя элемента: **`serve_path`**
Содержится в: `config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к программе serve
Может отсутствовать: *нет*
Может повторяться: *нет*
Используется в: super-serve
Описание. Данный элемент задаёт путь к программе serve системы ejudge.
Пример.

```
<serve_path>/usr/ejudge/bin/serve</serve_path>
```

2.4.12 Тег `run_path`

Имя элемента: **`run_path`**
Содержится в: `config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к программе run
Может отсутствовать: *нет*
Может повторяться: *нет*
Используется в: super-serve
Описание. Данный элемент задаёт путь к программе run системы ejudge.
Пример.

```
<serve_path>/usr/ejudge/bin/run</serve_path>
```

2.4.13 Тег `socket_path`

Имя элемента: **`socket_path`**
Содержится в: `config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к UNIX-сокету
Может отсутствовать: *нет*
Может повторяться: *нет*
Используется в: edit-userlist, userlist-server
Описание. Данный элемент задаёт путь к UNIX-сокету, который используется для связи с сервером информации о пользователях userlist-server. Сокет создаётся при старте этой программы, поэтому должен находится в каталоге, доступном ей на запись. Смотри также элемент `socket_path` конфигурационного файла `users.xml`, элемент `socket_path` конфигурационного файла `register.xml`, параметр `socket_path` конфигурационного файла `master.cfg`, параметр `socket_path` конфигурационного файла `judge.cfg`, параметр `socket_path` конфигурационного файла `team.cfg`, параметр `socket_path` конфигурационного файла `serve.cfg`.

Пример.

```
<socket_path>/var/ejudge/userlist-socket</socket_path>
```

2.4.14 Тег `user_map`

Имя элемента: **`user_map`**
Содержится в: `config`
Может содержать: `map`
Атрибуты: *нет*
Тип содержимого: игнорируется
Может отсутствовать: *да*
Может повторяться: *нет*
Используется в: userlist-server

Описание. Данный элемент является элементом верхнего уровня списка отображения системных (Unix) пользователей в пользователей ejudge. Используется для определения уровня привилегий пользователя при запуске им утилит edit-userlist и serve, когда последние подключаются к базе данных пользователей.

Пример.

```
<user_map>
  <map system_user="john" local_user="vasya_petrov"/>
</user_map>
```

2.4.15 Тег `userdb_file`

Имя элемента: **`userdb_file`**
Содержится в: `config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к файлу
Может отсутствовать: *нет*
Может повторяться: *нет*
Используется в: clean-users, userlist-server

Описание. Текст в данном элементе задаёт путь к файлу, хранящему информацию о пользователях системы `userlist.xml`.

Пример.

```
<userdb_file>/var/ejudge/db/userlist.xml</userdb_file>
```

2.5 Конфигурационный файл `register.xml`

Этот конфигурационный файл используется CGI-программой register для начальной инициализации после старта. По умолчанию конфигурационный файл должен находиться в каталоге `../cgi-data` относительно каталога, в котором находится исполняемый файл `register`. Изменение конфигурационного каталога возможно при компиляции программы в `makefile`. Если программа register запускается под другим именем, то имя конфигурационного файла будет соответствующим образом изменяться. Точные правила поиска конфигурационного файла приведены в разделе ???.

Например, если программа запускается под именем `register-`N, где N — некоторое десятичное число, то сначала будет сделана попытка считать конфигурационный файл `../cgi-data/register-`N`.xml`, затем, если эта попытка завершилась неудачно, будет сделана попытка считать конфигурационный файл `../cgi-data/register-`N'`.xml`, где N' — это число, напечатанное с 6 десятичными знаками, включая незначащие нули, например, `../cgi-data/register-000001.xml`. Если это попытка завершилась неудачно, будет сделана попытка считать конфигурационный файл `../cgi-data/register-`N''`.xml`, где N'' — число, напечатанное без ведущих незначащих нулей. Наконец, будет сделана попытка считать конфигурационный файл `../cgi-data/register.xml`, и в случае неуспеха программа сгенерирует сообщение об ошибке и завершится.

Конфигурационный файл представляет собой правильный XML-файл, элементом верхнего уровня которого является `register_config`.

Общая структура файла приведена ниже.

```
register_config
    access
        ip
    contests_dir
    l10n_dir
    socket_path
```

Далее даётся описание всех элементов конфигурационного файла.

2.5.1 Тег `register_config`

Имя элемента: **`register_config`**
Содержится в: *нет*
Может содержать: `access`, `contests_dir`, `l10n_dir`, `socket_path`
Атрибуты: `l10n`, `charset`, `show_generation_time`
Тип содержимого: игнорируется
Может отсутствовать: *нет*
Может повторяться: *нет*
Описание. Тег верхнего уровня конфигурационного файла.

Атрибут `l10n`

Имя атрибута: **`l10n`**
Содержится в: `register_config`
Тип значения: *boolean*
Значение по умолчанию: *false*
Может отсутствовать: *да*

Описание. Данный атрибут включает поддержку локализации интерфейса. Если его значение установлено в *true*, CGI-программа register считывает значение параметра `locale_id` и переключает язык интерфейса в зависимости от значения параметра. Подробнее вопросы локализации системы рассматриваются в разделе ???.

Логическое значение *true* задаётся с помощью строк `1`, `yes` или `true`, а логическое значение *false* — с помощью строк `0`, `no`, `false`.

Атрибут `charset`

Имя атрибута: **`charset`**
Содержится в: `register_config`
Тип значения: *string*
Значение по умолчанию: `iso8859-1`
Может отсутствовать: *да*

Описание. Данный атрибут позволяет задавать кодировку, которая будет проставляться в заголовке всех генерируемых страниц. Обратите внимание, что в текущей версии кодировка проставляется вне зависимости от языка интерфейса.

Атрибут `show_generation_time`

Имя атрибута: **`show_generation_time`**
Содержится в: `register_config`
Тип значения: *boolean*
Значение по умолчанию: `false`
Может отсутствовать: *да*

Описание. Если данный атрибут установлен в *true*, генерируемые html-страницы будут содержать информацию о процессорном времени в миллисекундах, затраченном на генерацию каждой страницы. Например, если язык интерфейса — английский, сообщение будет иметь примерно следующий вид:

```
Page generation time: 37 msec
```

2.5.2 Тег `access`

Имя элемента: **`access`**
Содержится в: `register_config`
Может содержать: `ip`
Атрибуты: `default`
Тип содержимого: игнорируется
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задавать ограничения на IP-адреса, с которых производится обращение к CGI-программе register. Ограничения, заданные в этом конфигурационном файле, проверяются до вывода регистрационной страницы и действуют на все турниры, определённые в системе. Эти ограничения могут использоваться для запрещения любого доступа к системе от «нежелательных» IP-адресов. Каждый турнир может накладывать дополнительные ограничения. Они задаются с помощью элемента `register_access` конфигурационного файла `contest.xml`. См. подробное описание ограничений IP-адреса в разделе 2.2.

Атрибут `default`

Имя атрибута: **`default`**
Содержится в: `access`
Тип значения: *allow* или *deny*
Значение по умолчанию: *deny*
Может отсутствовать: *да*

Описание. Данный атрибут задаёт права доступа в случае, если IP-адрес клиента не перечислен в списке IP-адресов. Если атрибут установлен в *allow*, такие IP-адреса считаются допустимыми, а если атрибут установлен в *deny*, доступ с таких IP-адресов запрещается. См. подробное описание ограничений IP-адреса в разделе 2.2.

2.5.3 Тег `ip`

Имя элемента: **`ip`**
Содержится в: `access`
Может содержать: *нет*
Атрибуты: `allow`
Тип содержимого: шаблон IP-адреса
Может отсутствовать: *да*
Может повторяться: *да*

Описание. Данный элемент определяет права доступа для клиентов с заданным шаблоном IP-адреса. Если атрибут `allow` установлен в *true*, таким клиентам разрешается работа с CGI-программой register. См. подробное описание ограничений IP-адреса в разделе 2.2.

Атрибут `allow`

Имя атрибута: **`allow`**
Содержится в: `access`
Тип значения: *boolean*
Значение по умолчанию: *false*
Может отсутствовать: *да*

Описание. Данный атрибут задаёт права доступа в случае, если IP-адрес клиента удовлетворяет спецификации IP-адреса в данном элементе. Если значение атрибута равно *true*, такой IP-адрес считаются допустимым, а если значение атрибут установлено в *false*, доступ с этого IP-адреса запрещается. См. подробное описание ограничений IP-адреса в разделе 2.2.

2.5.4 Тег `l10n_dir`

Имя элемента: **`l10n_dir`**
Содержится в: `register_config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к каталогу
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный элемент задаёт путь к каталогу, в котором хранятся локализованные сообщения. Для локализации сообщений используется GNU gettext. См. раздел ??? для информации о локализации системы.

Пример.

```
<l10n_dir>/usr/share/ejudge/locale</l10n_dir>
```

2.5.5 Тег `contests_dir`

Имя элемента: **`contests_dir`**
Содержится в: `register_config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к каталогу
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный элемент задаёт путь к каталогу, в котором хранятся описания турниров `contest.xml`. Каждое описание турнира находится в отдельном файле с именем, равным шестизначному номеру турнира, например, `000000.xml`.

Пример.

```
<contests_dir>/var/ejudge/contests</contests_dir>
```

2.5.6 Тег `socket_path`

Имя элемента: **`socket_path`**
Содержится в: `register_config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к UNIX-сокету
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный элемент задаёт путь к UNIX-сокету, который используется для связи с сервером информации о пользователях userlist-server. Когда программа register стартует, UNIX-сокет должен существовать и быть доступным для пользователя, от имени которого запускается программа. Как правило, это пользователь, от имени которого работает WEB-сервер.

Пример.

```
<socket_path>/var/ejudge/userlist-socket</socket_path>
```

2.6 Конфигурационный файл `users.xml`

Этот конфигурационный файл используется CGI-программой users для начальной инициализации после старта. По умолчанию конфигурационный файл должен находиться в каталоге `../cgi-data` относительно каталога, в котором находится исполняемый файл `users`. Изменение конфигурационного каталога возможно при компиляции программы в `makefile`. Если программа **users** запускается под другим именем, то имя конфигурационного файла будет соответствующим образом изменяться. Точные правила поиска конфигурационного файла приведены в разделе ???.

Например, если программа запускается под именем `users-`N, где N — некоторое десятичное число, то сначала будет сделана попытка считать конфигурационный файл `../cgi-data/users-`N`.xml`, затем, если эта попытка завершилась неудачно, будет сделана попытка считать конфигурационный файл `../cgi-data/users-`N'`.xml`, где N' — это число, напечатанное с 6 десятичными знаками, включая незначащие нули, например, `../cgi-data/users-000001.xml`. Если это попытка завершилась неудачно, будет сделана попытка считать конфигурационный файл `../cgi-data/users-`N''`.xml`, где N'' — число, напечатанное без ведущих незначащих нулей. Наконец, будет сделана попытка считать конфигурационный файл `../cgi-data/users.xml`, и в случае неуспеха программа сгенерирует сообщение об ошибке и завершится.

Конфигурационный файл представляет собой правильный XML-файл, элементом верхнего уровня которого является `users_config`.

Общая структура файла приведена ниже.

```
users_config
    access
        ip
    contests_dir
    l10n_dir
    socket_path
```

Далее даётся описание всех элементов конфигурационного файла.

2.6.1 Тег `users_config`

Имя элемента: **`users_config`**
Содержится в: *нет*
Может содержать: `access`, `contests_dir`, `l10n_dir`, `socket_path`
Атрибуты: `l10n`, `charset`, `show_generation_time`
Тип содержимого: игнорируется
Может отсутствовать: *нет*
Может повторяться: *нет*
Описание. Тег верхнего уровня конфигурационного файла.

Атрибут `l10n`

Имя атрибута: **`l10n`**
Содержится в: `users_config`
Тип значения: *boolean*
Значение по умолчанию: *false*
Может отсутствовать: *да*

Описание. Данный атрибут включает поддержку локализации интерфейса. Если его значение установлено в *true*, CGI-программа users считывает значение параметра `locale_id` и переключает язык интерфейса в зависимости от значения параметра. Подробнее вопросы локализации системы рассматриваются в разделе ???.

Логическое значение *true* задаётся с помощью строк `1`, `yes` или `true`, а логическое значение *false* — с помощью строк `0`, `no`, `false`.

Атрибут `charset`

Имя атрибута: **`charset`**
Содержится в: `users_config`
Тип значения: *string*
Значение по умолчанию: `iso8859-1`
Может отсутствовать: *да*

Описание. Данный атрибут позволяет задавать кодировку, которая будет проставляться в заголовке всех генерируемых страниц. Обратите внимание, что в текущей версии кодировка проставляется вне зависимости от языка интерфейса.

Атрибут `show_generation_time`

Имя атрибута: **`show_generation_time`**
Содержится в: `users_config`
Тип значения: *boolean*
Значение по умолчанию: `false`
Может отсутствовать: *да*

Описание. Если данный атрибут установлен в *true*, генерируемые html-страницы будут содержать информацию о процессорном времени в миллисекундах, затраченном на генерацию каждой страницы. Например, если язык интерфейса — английский, сообщение будет иметь примерно следующий вид:

```
Page generation time: 37 msec
```

2.6.2 Тег `access`

Имя элемента: **`access`**
Содержится в: `users_config`
Может содержать: `ip`
Атрибуты: `default`
Тип содержимого: игнорируется
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задавать ограничения на IP-адреса, с которых производится обращение к CGI-программе **users**. Ограничения, заданные в этом конфигурационном файле, проверяются до вывода регистрационной страницы и действуют на все турниры, определённые в системе. Эти ограничения могут использоваться для запрещения любого доступа к системе от «нежелательных» IP-адресов. Каждый турнир может накладывать дополнительные ограничения. Они задаются с помощью элемента `users_access` конфигурационного файла `contest.xml`. См. подробное описание ограничений IP-адреса в разделе 2.2.

Атрибут `default`

Имя атрибута: **`default`**
Содержится в: `access`
Тип значения: *allow* или *deny*
Значение по умолчанию: *deny*
Может отсутствовать: *да*

Описание. Данный атрибут задаёт права доступа в случае, если IP-адрес клиента не перечислен в списке IP-адресов. Если атрибут установлен в *allow*, такие IP-адреса считаются допустимыми, а если атрибут установлен в *deny*, доступ с таких IP-адресов запрещается. См. подробное описание ограничений IP-адреса в разделе 2.2.

2.6.3 Тег `ip`

Имя элемента: **`ip`**
Содержится в: `access`
Может содержать: *нет*
Атрибуты: `allow`
Тип содержимого: шаблон IP-адреса
Может отсутствовать: *да*
Может повторяться: *да*

Описание. Данный элемент определяет права доступа для клиентов с заданным шаблоном IP-адреса. Если атрибут `allow` установлен в *true*, таким клиентам разрешается работа с CGI-программой **users**. См. подробное описание ограничений IP-адреса в разделе 2.2.

Атрибут `allow`

Имя атрибута: **`allow`**
Содержится в: `access`
Тип значения: *boolean*
Значение по умолчанию: *false*
Может отсутствовать: *да*

Описание. Данный атрибут задаёт права доступа в случае, если IP-адрес клиента удовлетворяет спецификации IP-адреса в данном элементе. Если значение атрибута равно *true*, такой IP-адрес считаются допустимым, а если значение атрибут установлено в *false*, доступ с этого IP-адреса запрещается. См. подробное описание ограничений IP-адреса в разделе 2.2.

2.6.4 Тег `l10n_dir`

Имя элемента: **`l10n_dir`**
Содержится в: `users_config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к каталогу
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный элемент задаёт путь к каталогу, в котором хранятся локализованные сообщения. Для локализации сообщений используется GNU gettext. См. раздел ??? для информации о локализации системы.

Пример.

```
<l10n_dir>/usr/share/ejudge/locale</l10n_dir>
```

2.6.5 Тег `contests_dir`

Имя элемента: **`contests_dir`**
Содержится в: `users_config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к каталогу
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный элемент задаёт путь к каталогу, в котором хранятся описания турниров `contest.xml`. Каждое описание турнира находится в отдельном файле с именем, равным шестизначному номеру турнира, например, `000000.xml`.

Пример.

```
<contests_dir>/var/ejudge/contests</contests_dir>
```

2.6.6 Тег `socket_path`

Имя элемента: **`socket_path`**
Содержится в: `users_config`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к UNIX-сокету
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный элемент задаёт путь к UNIX-сокету, который используется для связи с сервером информации о пользователях userlist-server. Когда программа users стартует, UNIX-сокет должен существовать и быть доступным для пользователя, от имени которого запускается программа. Как правило, это пользователь, от имени которого работает WEB-сервер.

Пример.

```
<socket_path>/var/ejudge/userlist-socket</socket_path>
```

2.7 Конфигурационный файл `master.cfg`

Данный конфигурационный файл используется CGI-утилитой master. По умолчанию он должен находиться в каталоге `../cgi-data` относительно каталога, из которого запускается программа master. Изменение каталога по умолчанию возможно при компиляции системы ejudge изменением соответствующей переменной настройки в `makefile`. При запуске программы master она пытается считать конфигурационный файл из конфигурационного файла, причём имя конфигурационного файла может зависеть от номера турнира и имени, под которым запускается программа master. Точные правила поиска конфигурационного файла даны в разделе ???. Конфигурационный файл, считываемый, если никакой другой конфигурационный файл считать не удалось, называется `master.cfg`. Если и чтение этого файла завершилось с ошибкой, программа master генерирует сообщение ошибки конфигурации системы. Далее любой конфигурационный файл, используемый программой master будет называться `master.cfg`.

Конфигурационный файл `master.cfg` имеет текстовый формат, напоминающий формат `.ini`-файлов. Каждый конфигурационный параметр задаётся в виде

```
<имя параметра> = <значение>
```

где `<имя параметра>` может состоять из латинских заглавных и строчных букв, цифр и знака подчёркивания. `<значение>` должно заключаться в кавычки, если оно содержит пробельные символы, но может быть задано и без кавычек, если оно состоит из единственного слова. Пробельные символы в начале и конце строки и вокруг знака `="` игнорируются. Комментарии в конфигурационном файле начинаются с символов `#` или `;` и продолжаются до конца строки.

Параметры конфигурационного файла перечислены ниже.

2.7.1 Параметр `allow_deny`

Имя параметра: **`allow_deny`**
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*
Описание: См. раздел 2.2.5
Пример: См. раздел 2.7.3

2.7.2 Параметр `allow_from`

Имя параметра: **`allow_from`**
Тип содержимого: *спецификация IP-ограничений*
Может отсутствовать: *да*
Значение по умолчанию: пустая строка
Может повторяться: *нет*
Описание: См. раздел 2.2.5
Пример: См. раздел 2.7.3

2.7.3 Параметр `deny_from`

Имя параметра: **`deny_from`**
Тип содержимого: *спецификация IP-ограничений*
Может отсутствовать: *да*
Значение по умолчанию: пустая строка
Может повторяться: *нет*
Описание: См. раздел 2.2.5
Пример.

```
allow_deny = 1
allow_from = "192.168.0.1 192.168.1."
deny_from = "192.168."
```

2.7.4 Параметр `charset`

Имя параметра: **`charset`**
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `iso8859-1`
Может повторяться: *нет*
Описание. Данный параметр задаёт название кодировки, которое будет помещаться в http-заголовок генерируемой страницы. Полная информация о локализации системы ejudge приведена в разделе 2.3.3.
Пример.

```
charset = "koi8-r"
```

2.7.5 Параметр `enable_l10n`

Имя параметра: **`enable_l10n`**
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*
Описание: См. раздел 2.3.1
Пример: См. раздел 2.7.6
Замечание: В текущей версии программы master данный параметр не имеет никакого эффекта.

2.7.6 Параметр `l10n_dir`

Имя параметра: **`l10n_dir`**
Тип содержимого: путь к каталогу
Может отсутствовать: *да*
Значение по умолчанию: пустая строка
Может повторяться: *нет*
Описание: См. раздел 2.3.1
Замечание: В текущей версии программы master данный параметр не имеет никакого эффекта.

Если параметр `enable_l10n` установлен в *true*, а параметр `l10n_dir` не задан, локализация сообщений принудительно отключается.

Пример.

```
enable_l10n
l10n_dir = "/usr/share/locale"
```

2.7.7 Параметр `socket_path`

Имя параметра: **`socket_path`**
Тип содержимого: путь к UNIX-сокету
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный параметр задаёт путь к UNIX-сокету, который используется для связи с программой-сервером базы пользователей userlist-server. В момент запуска программы master сокет должен существовать.

Пример.

```
<socket_path>/var/ejudge/userlist-socket</socket_path>
```

2.7.8 Параметр `contests_dir`

Имя параметра: `contests_dir`
Тип содержимого: путь к каталогу
Может отсутствовать: *нет*
Может повторяться: *нет*
Описание. Данный элемент задаёт путь к каталогу, в котором хранятся описания турниров `contest.xml`. Каждое описание турнира находится в отдельном файле с именем, равным шестизначному номеру турнира, например, `000000.xml`.
Пример.

```
contests_dir = "/var/ejudge/contests"
```

2.8 Конфигурационный файл `judge.cfg`

Данный конфигурационный файл используется CGI-утилитой judge. По умолчанию он должен находиться в каталоге `../cgi-data` относительно каталога, из которого запускается программа judge. Изменение каталога по умолчанию возможно при компиляции системы ejudge изменением соответствующей переменной настройки в `makefile`. При запуске программы judge она пытается считать конфигурационный файл из конфигурационного файла, причём имя конфигурационного файла может зависеть от номера турнира и имени, под которым запускается программа judge. Точные правила поиска конфигурационного файла даны в разделе ???. Конфигурационный файл, считываемый, если никакой другой конфигурационный файл считать не удалось, называется `judge.cfg`. Если и чтение этого файла завершилось с ошибкой, программа judge генерирует сообщение ошибки конфигурации системы. Далее любой конфигурационный файл, используемый программой judge будет называться `judge.cfg`.

Формат конфигурационного файла `judge.cfg` в точности совпадает с форматом конфигурационного файла `master.cfg`, так как программы master и judge — это одна программа, запускаемая под разными именами. Конфигурационный файл `judge.cfg` имеет текстовый формат, напоминающий формат `.ini`-файлов. Каждый конфигурационный параметр задаётся в виде

```
<имя параметра> = <значение>
```

где `<имя параметра>` может состоять из латинских заглавных и строчных букв, цифр и знака подчёркивания. `<значение>` должно заключаться в кавычки, если оно содержит пробельные символы, но может быть задано и без кавычек, если оно состоит из единственного слова. Пробельные символы в начале и конце строки и вокруг знака `="` игнорируются. Комментарии в конфигурационном файле начинаются с символов `#` или `;` и продолжаются до конца строки.

Параметры конфигурационного файла перечислены ниже.

2.8.1 Параметр `allow_deny`

Имя параметра: **`allow_deny`**
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*
Описание: См. раздел 2.2.5
Пример: См. раздел 2.8.3

2.8.2 Параметр `allow_from`

Имя параметра: **`allow_from`**
Тип содержимого: *спецификация IP-ограничений*
Может отсутствовать: *да*
Значение по умолчанию: пустая строка
Может повторяться: *нет*
Описание: См. раздел 2.2.5
Пример: См. раздел 2.8.3

2.8.3 Параметр `deny_from`

Имя параметра: **`deny_from`**
Тип содержимого: *спецификация IP-ограничений*
Может отсутствовать: *да*
Значение по умолчанию: пустая строка
Может повторяться: *нет*
Описание: См. раздел 2.2.5
Пример.

```
allow_deny = 1
allow_from = "192.168.0.1 192.168.1."
deny_from = "192.168."
```

2.8.4 Параметр `charset`

Имя параметра: **`charset`**
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `iso8859-1`
Может повторяться: *нет*
Описание. Данный параметр задаёт название кодировки, которое будет помещаться в http-заголовок генерируемой страницы. Полная информация о локализации системы ejudge приведена в разделе 2.3.3.
Пример.

```
charset = "koi8-r"
```

2.8.5 Параметр `enable_l10n`

Имя параметра: **`enable_l10n`**
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*
Описание: См. раздел 2.3.1
Пример: См. раздел 2.8.6
Замечание: В текущей версии программы judge данный параметр не имеет никакого эффекта.

2.8.6 Параметр `l10n_dir`

Имя параметра: **`l10n_dir`**
Тип содержимого: путь к каталогу
Может отсутствовать: *да*
Значение по умолчанию: пустая строка
Может повторяться: *нет*
Описание: См. раздел 2.3.1
Замечание: В текущей версии программы judge данный параметр не имеет никакого эффекта.

Если параметр `enable_l10n` установлен в *true*, а параметр `l10n_dir` не задан, локализация сообщений принудительно отключается.

Пример.

```
enable_l10n
l10n_dir = "/usr/share/locale"
```

2.8.7 Параметр `socket_path`

Имя параметра: **`socket_path`**
Тип содержимого: путь к UNIX-сокету
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный параметр задаёт путь к UNIX-сокету, который используется для связи с программой-сервером базы пользователей userlist-server. В момент запуска программы judge сокет должен существовать.

Пример.

```
socket_path = "/var/ejudge/userlist-socket"
```

2.8.8 Параметр `contests_dir`

Имя параметра: `contests_dir`
Тип содержимого: путь к каталогу
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный элемент задаёт путь к каталогу, в котором хранятся описания турниров `contest.xml`. Каждое описание турнира находится в отдельном файле с именем, равным шестизначному номеру турнира, например, `000000.xml`.

Пример.

```
contests_dir = "/var/ejudge/contests"
```

2.9 Конфигурационный файл `team.cfg`

Данный конфигурационный файл используется CGI-утилитой team. По умолчанию он должен находиться в каталоге `../cgi-data` относительно каталога, из которого запускается программа team. Изменение каталога по умолчанию возможно при компиляции системы ejudge изменением соответствующей переменной настройки в `makefile`. При запуске программы team она пытается считать конфигурационный файл из конфигурационного файла, причём имя конфигурационного файла может зависеть от номера турнира и имени, под которым запускается программа team. Точные правила поиска конфигурационного файла даны в разделе ???. Конфигурационный файл, считываемый, если никакой другой конфигурационный файл считать не удалось, называется `team.cfg`. Если и чтение этого файла завершилось с ошибкой, программа team генерирует сообщение ошибки конфигурации системы. Далее любой конфигурационный файл, используемый программой team будет называться `team.cfg`.

Формат конфигурационного файла `team.cfg` практически совпадает с форматом конфигурационного файла `master.cfg`. Конфигурационный файл `team.cfg` имеет текстовый формат, напоминающий формат `.ini`-файлов. Каждый конфигурационный параметр задаётся в виде

```
<имя параметра> = <значение>
```

где `<имя параметра>` может состоять из латинских заглавных и строчных букв, цифр и знака подчёркивания. `<значение>` должно заключаться в кавычки, если оно содержит пробельные символы, но может быть задано и без кавычек, если оно состоит из единственного слова. Пробельные символы в начале и конце строки и вокруг знака `="` игнорируются. Комментарии в конфигурационном файле начинаются с символов `#` или `;` и продолжаются до конца строки.

Параметры конфигурационного файла перечислены ниже.

2.9.1 Параметр `allow_deny`

Имя параметра: **`allow_deny`**
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*
Описание: См. раздел 2.2.5
Пример: См. раздел 2.9.3

2.9.2 Параметр `allow_from`

Имя параметра: **`allow_from`**
Тип содержимого: *спецификация IP-ограничений*
Может отсутствовать: *да*
Значение по умолчанию: пустая строка
Может повторяться: *нет*
Описание: См. раздел 2.2.5
Пример: См. раздел 2.9.3

2.9.3 Параметр `deny_from`

Имя параметра: **`deny_from`**
Тип содержимого: *спецификация IP-ограничений*
Может отсутствовать: *да*
Значение по умолчанию: пустая строка
Может повторяться: *нет*
Описание: См. раздел 2.2.5
Пример.

```
allow_deny = 1
allow_from = "192.168.0.1 192.168.1."
deny_from = "192.168."
```

2.9.4 Параметр `charset`

Имя параметра: **`charset`**
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `iso8859-1`
Может повторяться: *нет*
Описание. Данный параметр задаёт название кодировки, которое будет помещаться в http-заголовок генерируемой страницы. Полная информация о локализации системы ejudge приведена в разделе 2.3.3.
Пример.

```
charset = "koi8-r"
```

2.9.5 Параметр `enable_l10n`

Имя параметра: `enable_l10n`
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*
Описание: См. раздел 2.3.1
Пример: См. раздел 2.9.6

2.9.6 Параметр `l10n_dir`

Имя параметра: `l10n_dir`
Тип содержимого: путь к каталогу
Может отсутствовать: *да*
Значение по умолчанию: пустая строка
Может повторяться: *нет*
Описание: См. раздел 2.3.1

Если параметр `enable_l10n` установлен в *true*, а параметр `l10n_dir` не задан, локализация сообщений принудительно отключается.

Пример.

```
enable_l10n
l10n_dir = "/usr/share/locale"
```

2.9.7 Параметр `socket_path`

Имя параметра: `socket_path`
Тип содержимого: путь к UNIX-сокету
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данный параметр задаёт путь к UNIX-сокету, который используется для связи с программой-сервером базы пользователей userlist-server. В момент запуска программы team сокет должен существовать.

Пример.

```
socket_path = "/var/ejudge/userlist-socket"
```

2.9.8 Параметр `contests_dir`

Имя параметра: **`contests_dir`**
Тип содержимого: путь к каталогу
Может отсутствовать: *нет*
Может повторяться: *нет*
Описание. Данный элемент задаёт путь к каталогу, в котором хранятся описания турниров `contest.xml`. Каждое описание турнира находится в отдельном файле с именем, равным шестизначному номеру турнира, например, `000000.xml`.
Пример.

```
contests_dir = "/var/ejudge/contests"
```

2.9.9 Параметр `show_generation_time`

Имя параметра: **`show_generation_time`**
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*
Описание. Если данный параметр установлен в *true*, в конце генерируемой html-страницы добавляется информация о процессорном времени, затраченном на её генерацию.
Пример.

```
show_generation_time
```

2.10 Конфигурационный файл турниров `contest.xml`

В данном разделе даётся описание формата файла описания параметров турнира. В системе ejudge описание каждого турнира хранится в отдельном XML-файле. Все файлы располагаются в одном каталоге, который задаётся в элементе `contests_dir` для конфигурационных файлов в формате XML или в параметре `contests_dir` для конфигурационных файлов в формате INI. Описание каждого турнира хранится в этом каталоге в файле с именем N`.xml`, где N — идентификатор (номер) турнира в десятичной записи из 6 знаков с ведущими нулями. Например, информация о турнире с номером 1 хранится в файле `000001.xml`, а информация о турнире с номером 192 — в файле `000192.xml`. Идентификатор турнира назначается администратором системы ejudge и должен быть целым числом в диапазоне $1 \leqslant N \leqslant 999999$. Никакие два турнира в одной инсталляции системы ejudge не могут иметь один и тот же идентификатор. Для большей эффективности работы системы ejudge рекомендуется нумеровать турниры подряд, начиная с 1. Идентификатор турнира, определяемый по имени файла, в котором хранится его описание, должен совпадать со значением атрибута `id` элемента верхнего уровня `contest` самого файла.

Файл описания турнира должен представлять собой правильно сформированный XML-файл. Элементом верхнего уровня файла является элемент `contest`, который должен присутствовать и не может повторяться. Общая структура вложенности элементов файла приведена в таблице 2.3.

В конфигурационном файле турнира, в частности, задаются дополнительные ограничения на допустимые IP-адреса для всех CGI-программ (элементы ...). Кроме этого, в конфигурационном файле определяются обязательные и необязательные поля анкеты участника.

Анкета участника турнира заполняется пользователем, когда он регистрируется на турнир с помощью CGI-программы register. Участником турнира может быть как команда (для командных соревнований), так и единственный человек. Поэтому анкета участника турнира состоит из нескольких частей. Первую часть составляют данные общего характера, такие как название участника турнира (то есть название команды или имя, под которым в турнире выступает человек), учебное заведение, факультет, город, страна. Вторую часть анкеты составляют персональные данные всех лиц, связанных с командой (в случае командного турнира) или участником. Для командных турниров могут задаваться личные данные членов команды, запасных игроков, тренеров и руководителей команды, а также гостей, приглашаемых командой. Для личных турниров могут задаваться личные данные участника турнира, его тренера и руководителя.

Поскольку количество требуемой информации об участнике может варьироваться от турнира к турниру, так же как и ограничения на число лиц, связанных с участником или командой, в конфигурационном файле турнира задаются поля анкеты, предъявляемые для заполнения. Поля анкеты первой части описываются в элементах `field`, вложенных непосредственно в элемент `contest`. Атрибут `id` элемента в этом случае может принимать значения, перечисленные в таблице 2.4.

Поля второй части анкеты описываются отдельно по каждой категории лиц, связанных с участником. Выделяются пять категорий лиц:

1. Непосредственно игроки (`contestants`). В случае командного турнира это — члены команды, в случае личного турнира это сам участник турнира.

2. Запасные игроки (`reserves`). В случае личного турнира эта категория не имеет смысла.

```
contest
      name
      name_en
      register_access
            ip
      users_access
            ip
      master_access
            ip
      judge_access
            ip
      team_access
            ip
      field
      contestants
            field
      reserves
            field
      coaches
            field
      advisors
            field
      guests
            field
      users_header_file
      users_footer_file
      register_header_file
      register_footer_file
      team_header_file
      team_footer_file
      register_email
      register_url
      team_url
      registration_deadline
      users_head_style
      users_par_style
      users_table_style
      users_verb_style
      register_head_style
      register_par_style
      register_table_style
      team_head_style
      team_par_style

      caps
            cap
      root_dir
      standings_url
      problems_url
      client_flags
      serve_user
      serve_group
```

Таблица 2.3: Структура вложенности элементов конфигурационного файла `contest.xml`

homepage	Домашняя страница пользователя.
inst	Полное название учебного заведения, к которому относится участник.
inst_en	Полное название учебного заведения, к которому относится участник, на английском языке.
instshort	Краткое название учебного заведения.
instshort_en	Краткое название учебного заведения на английском языке.
fac	Полное название факультета, к которому относится участник.
fac_en	Полное название факультета, к которому относится участник, на английском языке.
facshort	Краткое название факультета.
facshort_en	Краткое название факультета на английском языке.
city	Город.
city_en	Название города на английском языке.
country	Страна.
country_en	Название страны на английском языке.

Таблица 2.4: Описание полей анкеты участника, допустимых в первой части анкеты

3. Тренеры (coaches).
4. Руководители (advisors).
5. Прочие (guests).

Для каждой из категории атрибутами соответствующего элемента файла конфигурации турнира задаётся минимальное и максимальное число лиц этой категории. Поля анкеты задаются с помощью вложенных элементов field, атрибут id которых может принимать значения, перечисленные в таблице 2.5.

Далее рассматриваются все элементы конфигурационного файла contest.xml.

firstname	Имя.
firstname_en	Имя в английском написании.
middlename	Отчество.
middlename_en	Отчество в английском написании.
surname	Фамилия.
surname_en	Фамилия в английском написании.
status	Статус: школьник, студент, аспирант, учитель и т. д.
grade	Класс школы или номер курса ВУЗа.
group	Название класса или академической группы.
group_en	Название класса или академической группы в английском написании.
email	Личный адрес e-mail.
homepage	Личная страничка в Интернет.
inst	Полное название учебного заведения (если отличается от заданного для участника в целом).
inst_en	Полное название учебного заведения (если отличается от заданного для участника в целом) на английском языке.
instshort	Краткое название учебного заведения.
instshort_en	Краткое название учебного заведения на английском языке.
fac	Полное название факультета.
fac_en	Полное название факультета на английском языке.
facshort	Краткое название факультета.
facshort_en	Краткое название факультета на английском языке.
occupation	Занимаемая должность.
occupation_en	Занимаемая должность на английском языке.

Таблица 2.5: Описание полей анкеты участника, допустимых во второй части анкеты

2.10.1 Элемент `contest`

Имя элемента: **`contest`**
Содержится в: *нет*
Может содержать: `name`, `name_en`, `register_access`, `users_access`, `master_access`, `judge_access`, `team_access`, `field`, `contestants`, `reserves`, `coaches`, `advisors`, `guests`, `users_header_file`, `users_footer_file`, `register_header_file`, `register_footer_file`, `team_header_file`, `team_footer_file`, `register_email`, `register_url`, `team_url`, `registration_deadline`, `caps`, `root_dir`, `standings_url`, `problems_url`, `client_flags`, `serve_user`, `serve_group`, `users_head_style`, `users_par_style`, `users_table_style`, `users_verb_style`, `register_head_style`, `register_par_style`, `register_table_style`, `team_head_style`, `team_par_style`
Атрибуты: `id`, `autoregister`, `managed`, `run_managed`, `clean_users`, `disable_team_password`
Тип содержимого: игнорируется
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Элемент верхнего уровня конфигурационного файла. Элемент должен иметь обязательный атрибут `id`, определяющий идентификатор этого турнира.

Атрибут `id`

Имя атрибута: **`id`**
Содержится в: `contest`
Тип значения: *integer*
Может отсутствовать: *нет*

Описание. Данный атрибут должен содержать идентификатор турнира. Идентификатор турнира представляет собой число в десятичной записи в диапазоне от 1 до 999999 включительно. Идентификатор турнира, определяемого в файле, должен соответствовать имени этого файла.

Атрибут `autoregister`

Имя атрибута: **`autoregister`**
Содержится в: `contest`
Тип значения: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*

Описание. Данный атрибут устанавливает значение флага авторегистрации. Если атрибут установлен в значение *true*, регистрация каждого нового участника турнира автоматически подтверждается как только он полностью заполнил регистрационную форму. Сразу после этого участник может заходить на личную страницу сдачи решений. Если атрибут установлен в значение *false* (по умолчанию), после того, как участник полностью заполнил регистрационную форму, его регистрации присваивается статус «Ожидает подтверждения». В этом статусе участник не может заходить на свою личную страницу. Подтверждение регистрации может быть сделано привилегированным пользователем турнира, который имеет полномочие `EDIT_REGISTRATION`.

Значение этого атрибута рекомендуется устанавливать в *true* для постоянно открытых турниров, таких как интернет-туры или виртуальные турниры. Значение атрибута *false* позволяет регулировать регистрацию на турнир и может применяться для традиционной формы проведения турниров.

Атрибут `managed`

Имя атрибута: **`managed`**
Содержится в: `contest`
Тип значения: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*

Описание. Если значение этого атрибута установлено в *true*, программа управления турниром serve ставится под управление программы super-serve, которая следит одновременно за несколькими турнирами и запускает программу serve, как только поступает запрос соответствующему турниру. Если значение атрибута установлено в *false*, данный турнир игнорируется программой super-serve.

Атрибут `run_managed`

Имя атрибута: **`run_managed`**
Содержится в: `contest`
Тип значения: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*

Описание. Если значение этого атрибута установлено в *true*, программа проверки решений run ставится под управление программы super-serve, которая следит одновременно за несколькими турнирами и запускает программу run, как только поступает запрос на проверку решения в соответствующем турнире. Если значение атрибута установлено в *false*, данный турнир игнорируется программой super-serve.

Атрибут `clean_users`

Имя атрибута: `clean_users`
Содержится в: `contest`
Тип значения: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *true*

Описание. Если значение данного атрибута установлено в *true* (значение по умолчанию), программа clean-users будет сканировать базу сдач решений и вопросов судьям текущего турнира. Программа clean-users выявляет пользователей, занесённых в базу пользователей, возможно зарегистрировавшихся для участия в каком-либо турнире, но не сделавших ни одной попытки сдачи решения и не задавших ни один вопрос. Такие пользователи помечаются как кандидаты на удаление из базы пользователей. Если значение данного атрибута установлено в *false*, программа clean-users игнорирует данный турнир.

Атрибут `disable_team_password`

Имя атрибута: `disable_team_password`
Содержится в: `contest`
Тип значения: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*

Описание. По умолчанию каждый пользователь имеет два пароля: регистрационный и пароль участника. Регистрационный пароль необходим для доступа к CGI-программе register, которая позволяет изменять регистрационную информацию и регистрироваться для участия в турнирах. Пароль участника необходим для доступа к CGI-программе team, то есть непосредственно для участия в турнире. Если значение атрибута `disable_team_password` равно *true*, специальный пароль участия в турнире отключается. Для доступа к CGI-программе team в этом случае используется регистрационный пароль.

Пароли участников обычно генерируются администратором турнира (точнее, пользователем с полномочием `GENERATE_TEAM_PASSWORDS`) перед началом турнира. Для турниров с открытой регистрацией (см. атрибут `autoregister` элемента `contest`) это становится затруднительным. Тогда использование атрибута `disable_team_password` может быть удобным.

2.10.2 Элемент `name`

Имя элемента: **`name`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: *string*
Может отсутствовать: *нет*
Может повторяться: *нет*
Описание. Данный элемент позволяет задать имя турнира.
Пример.

```
<name>Турнир пионеров</name>
```

2.10.3 Элемент `name_en`

Имя элемента: **`name`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: *string*
Может отсутствовать: *да*
Может повторяться: *нет*
Описание. Данный элемент позволяет задать имя турнира на английском языке. Английское имя турнира используется программами users, register и team когда идентификатор языка (`locale_id`) равен 0. Если элемент `name_en` не задан, используется значение элемента `name`, то есть имя турнира на локальном языке (в настоящее время это может быть только русский язык).

Замечание. Английское название турнира должно использоваться во всех случаях, когда идентификатор языка в CGI-программах не совпадает с идентификатором языка, указанным в конфигурационном файле турнира. Данная возможность в настоящее время не реализована, поскольку система ejudge поддерживает пока только два языка: английский и русский.

Пример.

```
<name_en>The pioneer's tournament</name_en>
```

2.10.4 Элемент `register_access`

Имя элемента: **`register_access`**
Содержится в: `contest`
Может содержать: `ip`
Атрибуты: `default`
Тип содержимого: *список IP-ограничений*
Может отсутствовать: *да*
Может повторяться: *нет*
Описание. Данный элемент позволяет задать ограничения на IP-адреса, с которых возможна регистрация на данный турнир с помощью CGI-программы register. Для подробного описания средств ограничения доступа по IP-адресам системы ejudge см. раздел 2.2.

2.10.5 Элемент `users_access`

Имя элемента: `users_access`
Содержится в: `contest`
Может содержать: `ip`
Атрибуты: `default`
Тип содержимого: *список IP-ограничений*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать ограничения на IP-адреса, с которых возможен просмотр списка участников и информации об участниках с помощью CGI-программы users. Для подробного описания средств ограничения доступа по IP-адресам системы ejudge см. раздел 2.2.

2.10.6 Элемент `master_access`

Имя элемента: `master_access`
Содержится в: `contest`
Может содержать: `ip`
Атрибуты: `default`
Тип содержимого: *список IP-ограничений*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать ограничения на IP-адреса, с которых возможно администрирование данного турнира с помощью CGI-программы master. Для подробного описания средств ограничения доступа по IP-адресам системы ejudge см. раздел 2.2.

2.10.7 Элемент `judge_access`

Имя элемента: `judge_access`
Содержится в: `contest`
Может содержать: `ip`
Атрибуты: `default`
Тип содержимого: *список IP-ограничений*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать ограничения на IP-адреса, с которых возможно судейство данного турнира с помощью CGI-программы judge. Для подробного описания средств ограничения доступа по IP-адресам системы ejudge см. раздел 2.2.

2.10.8 Элемент `team_access`

Имя элемента: **`team_access`**
Содержится в: `contest`
Может содержать: `ip`
Атрибуты: `default`
Тип содержимого: *список IP-ограничений*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать ограничения на IP-адреса, с которых возможно участие в данном турнире с помощью CGI-программы team. Для подробного описания средств ограничения доступа по IP-адресам системы ejudge см. раздел 2.2.

2.10.9 Элемент `ip`

Имя элемента: **`ip`**
Содержится в: `register_access`, `users_access`, `master_access`, `judge_access`, `team_access`
Может содержать: *нет*
Атрибуты: `allow`
Тип содержимого: *IP-ограничение*
Может отсутствовать: *да*
Может повторяться: *да*

Описание. Данный элемент задаёт одно ограничения на IP-адреса. Для подробного описания средств ограничения доступа по IP-адресам системы ejudge см. раздел 2.2.

2.10.10 Элемент `field`

Имя элемента: **`field`**
Содержится в: `contest`, `contestants`, `reserves`, `coaches`, `advisors`, `guests`
Может содержать: *нет*
Атрибуты: `id`, `mandatory`
Тип содержимого: игнорируется
Может отсутствовать: *да*
Может повторяться: *да*

Описание. Элемент `field` позволяет указывать поля ввода, которые будут предъявлены участнику при регистрации на турнир. Если элемент находится непосредственно в элементе `contest`, он позволяет указывать поля ввода общей информации об участнике. В этом случае атрибут `id` должен принимать одно из значений, перечисленных в таблице 2.4. Если элемент находится в элементах `contestants`, `reserves`, `coaches`, `advisors`, и `guests`, с его помощью указываются поля анкеты, заполняемые для соответствующей категории лиц. В этом случае атрибут `id` должен принимать одно из значений, перечисленных в таблице 2.5. С помощью атрибута `mandatory` элемента `field` можно указать, что данное поле ввода является обязательным для заполнения в анкете.

Атрибут `id`

Имя атрибута: **`id`**
Содержится в: `field`
Тип значения: См. таблицы 2.4 и 2.5.
Может отсутствовать: *нет*

Описание. С помощью данного атрибута указывается поле ввода анкеты. Если элемент `field` непосредственно вложен в элемент `contest`, данный атрибут должен принимать одно из значений, перечисленных в таблице 2.4. Если элемент `field` вложен в элементы `contestants`, `reserves`, `coaches`, `advisors`, и `guests`, данный атрибут должен принимать одно из значений, перечисленных в таблице 2.5.

Атрибут `mandatory`

Имя атрибута: **`mandatory`**
Содержится в: `field`
Тип значения: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*

Описание. Данный атрибут позволяет указать, что поле ввода анкеты, задаваемое с помощью данного элемента `field`, обязательно к заполнению.

2.10.11 Элементы `contestants`, `reserves`, `coaches`, `advisors`, `guests`

Имя элемента:	**`contestants`**, **`reserves`**, **`coaches`**, **`advisors`**, **`guests`**
Содержится в:	`contest`
Может содержать:	`field`
Атрибуты:	`min`, `max`, `initial`
Тип содержимого:	игнорируется
Может отсутствовать:	*да*
Может повторяться:	*нет*

Описание. Описание нескольких элементов объединено в один раздел, так как эти элементы очень близки друг другу. Данные элементы позволяют задавать ограничения на количество лиц соответствующей категории. Кроме того, данные элементы содержат в себе элементы `field`, задающие поля ввода регистрационной анкеты. Элемент `contestants` соответствует категории игроков турнира, элемент `reserves` соответствует категории запасных игроков, элемент `coaches` — категории тренеров, элемент `advisors` — категории руководителей, и элемент `guests` — прочим.

Атрибут `min` позволяет задать минимальное число лиц в данной категории. Например, минимальное количество игроков одной команды (contestants) в командном турнире может быть установлено в 2 или 3, а в личном турнире — в 1. Если у каждого участника должен быть тренер, минимальное число лиц этой категории (coaches) должно быть установлено в 1. Атрибут `max` позволяет задать максимальное число лиц в данной категории. Например, максимальное количество игроков одной команды в командном турнире равно 3, а в личном турнире — 1. Атрибут `initial` позволяет задать, формы ввода для скольких лиц в данной категории будут выведены при регистрации на турнир, когда формы печатаются для данного пользователя первый раз.

Атрибут `min`

Имя атрибута:	**`min`**
Содержится в:	`contestants`, `reserves`, `coaches`, `advisors`, `guests`
Тип значения:	*integer*
Может отсутствовать:	*да*
Значение по умолчанию:	0
Описание:	Минимальное количество лиц в данной категории.

Атрибут `max`

Имя атрибута:	**`max`**
Содержится в:	`contestants`, `reserves`, `coaches`, `advisors`, `guests`
Тип значения:	*integer*
Может отсутствовать:	*да*
Значение по умолчанию:	0
Описание:	Максимальное количество лиц в данной категории.

Атрибут `initial`

Имя атрибута:	**`initial`**
Содержится в:	contestants, reserves, coaches, advisors, guests
Тип значения:	*integer*
Может отсутствовать:	*да*
Значение по умолчанию:	0
Описание:	Начальное количество лиц в данной категории.

Пример

```
<contestants min="2" max="3">
  <field id="firstname" mandatory="yes"/>
  <field id="middlename"/>
  <field id="surname" mandatory="yes"/>
  <field id="status" mandatory="yes"/>
  <field id="grade" mandatory="yes"/>
  <field id="group"/>
</contestants>
<reserves min="0" max="1">
  <field id="firstname" mandatory="yes"/>
  <field id="middlename"/>
  <field id="surname" mandatory="yes"/>
  <field id="status" mandatory="yes"/>
  <field id="grade" mandatory="yes"/>
  <field id="group"/>
</reserves>
<coaches min="1" max="1">
  <field id="firstname" mandatory="yes"/>
  <field id="middlename"/>
  <field id="surname" mandatory="yes"/>
  <field id="status" mandatory="yes"/>
  <field id="occupation" mandatory="yes"/>
</coaches>
<advisors min="0" max="1">
  <field id="firstname" mandatory="yes"/>
  <field id="middlename"/>
  <field id="surname" mandatory="yes"/>
  <field id="status" mandatory="yes"/>
  <field id="occupation" mandatory="yes"/>
</advisors>
```

2.10.12 Элемент `users_header_file`

Имя элемента: **`users_header_file`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к файлу
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать путь к файлу, который содержит «шапку» html-файла, генерируемого CGI-программой users. Если элемент задан, то содержимое указанного файла будет выведено в начале генерируемого файла. Если элемент не задан, генерируется заголовок по умолчанию, включающий название турнира и кодировку страницы. В файле-заголовке допускаются специальные конструкции, начинающиеся со знака `%` («процент»), перечисленные в таблице 2.6.

Пример.

```
<users_header_file>/home/judges/20/conf/header.shtml</users_header_file>
```

Замечание. Если в файле-заголовке используются server-side includes (SSI), поддержка SSI в CGI-программах должна быть включена в используемом http-сервере. Для дальнейшей информации обратитесь к документации на Ваш http-сервер.

2.10.13 Элемент `users_footer_file`

Имя элемента: **`users_footer_file`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к файлу
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать путь к файлу, который содержит «подвал» html-файла, генерируемого CGI-программой users. Если элемент задан, то содержимое указанного файла будет выведено в конце генерируемого файла. Если элемент не задан, генерируется подвал страницы по умолчанию, включающий информацию об авторских правах и версии системы ejudge. В файле-подвале допускаются специальные конструкции, начинающиеся со знака `%` («процент»), перечисленные в таблице 2.7.

Пример.

```
<users_footer_file>/home/judges/20/conf/footer.shtml</users_footer_file>
```

Замечание. Если в файле-подвале используются server-side includes (SSI), поддержка SSI в CGI-программах должна быть включена в используемом http-сервере. Для дальнейшей информации обратитесь к документации на Ваш http-сервер.

2.10.14 Элемент `register_header_file`

Имя элемента: `register_header_file`
Содержится в: contest
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к файлу
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать путь к файлу, который содержит «шапку» html-файла, генерируемого CGI-программой register. Если элемент задан, то содержимое указанного файла будет выведено в начале генерируемого файла. Если элемент не задан, генерируется заголовок по умолчанию, включающий название турнира и кодировку страницы. В файле-заголовке допускаются специальные конструкции, начинающиеся со знака % («процент»), перечисленные в таблице 2.6.

Пример.

```
<register_header_file>/home/judges/20/conf/header.shtml</register_header_file>
```

Замечание. Если в файле-заголовке используются server-side includes (SSI), поддержка SSI в CGI-программах должна быть включена в используемом http-сервере. Для дальнейшей информации обратитесь к документации на Ваш http-сервер.

2.10.15 Элемент `register_footer_file`

,

Имя элемента: `register_footer_file`
Содержится в: contest
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к файлу
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать путь к файлу, который содержит «подвал» html-файла, генерируемого CGI-программой register. Если элемент задан, то содержимое указанного файла будет выведено в конце генерируемого файла. Если элемент не задан, генерируется подвал страницы по умолчанию, включающий информацию об авторских правах и версии системы ejudge. В файле-подвале допускаются специальные конструкции, начинающиеся со знака % («процент»), перечисленные в таблице 2.7.

Пример.

```
<register_footer_file>/home/judges/20/conf/footer.shtml</register_footer_file>
```

Замечание. Если в файле-подвале используются server-side includes (SSI), поддержка SSI в CGI-программах должна быть включена в используемом http-сервере. Для дальнейшей информации обратитесь к документации на Ваш http-сервер.

2.10.16 Элемент `team_header_file`

Имя элемента: `team_header_file`
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к файлу
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать путь к файлу, который содержит «шапку» html-файла, генерируемого CGI-программой team. Если элемент задан, то содержимое указанного файла будет выведено в начале генерируемого файла. Если элемент не задан, генерируется заголовок по умолчанию, включающий название турнира и кодировку страницы. В файле-заголовке допускаются специальные конструкции, начинающиеся со знака `%` («процент»), перечисленные в таблице 2.6.

Пример.

```
<team_header_file>/home/judges/20/conf/header.shtml</team_header_file>
```

Замечание. Если в файле-заголовке используются server-side includes (SSI), поддержка SSI в CGI-программах должна быть включена в используемом http-сервере. Для дальнейшей информации обратитесь к документации на Ваш http-сервер.

2.10.17 Элемент `team_footer_file`

Имя элемента: `team_footer_file`
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: путь к файлу
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать путь к файлу, который содержит «подвал» html-файла, генерируемого CGI-программой team. Если элемент задан, то содержимое указанного файла будет выведено в конце генерируемого файла. Если элемент не задан, генерируется подвал страницы по умолчанию, включающий информацию об авторских правах и версии системы ejudge. В файле-подвале допускаются специальные конструкции, начинающиеся со знака `%` («процент»), перечисленные в таблице 2.7.

Пример.

```
<team_footer_file>/home/judges/20/conf/footer.shtml</team_footer_file>
```

Замечание. Если в файле-подвале используются server-side includes (SSI), поддержка SSI в CGI-программах должна быть включена в используемом http-сервере. Для дальнейшей информации обратитесь к документации на Ваш http-сервер.

2.10.18 Элемент `users_head_style`

`users_head_style`

Имя элемента: **`users_head_style`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: строка
Может отсутствовать: *да*
Значение по умолчанию: `h2`
Может повторяться: *нет*

Описание. Данный элемент позволяет установить тег языка HTML, который используется для выделения названий секций в html-странице, генерируемой CGI-программой users. Если значение этого элемента не установлено, используется тег `h2`, то есть заголовки в генерируемом html-документе будут иметь примерно следующий вид.

```
<h2>Список участников турнира X</h2>
```

Пример.

```
<users_head_style>h3</users_head_style>
```

Такое значение данной конфигурационной переменной приведёт к тому, что тот же самый заголовок будет иметь следующий вид.

```
<h3>Список участников турнира X</h3>
```

Замечание. В текущей реализации системы ejudge отсутствует возможность задать отдельно открывающий и закрывающий тег. И для того, и для другого будет использоваться значение данной переменной. Поэтому указание ещё каких-либо дополнительных атрибутов тега заголовка приведёт к тому, что дополнительные атрибуты продублируются в закрывающем теге, приводя, строго говоря, к неправильному html.

2.10.19 Элемент `users_par_style`

`users_par_style`

Имя элемента: **`users_par_style`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: строка
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данный элемент позволяет задать параметры тега `<p>` в html-страницах, генерируемых программой `users`. Значение элемента `users_par_style` добавляется в открывающий тег `<p>`. Закрывающий тег `</p>` остается без изменений.

Пример.

```
<users_par_style> class="common"</users_par_style>
```

Такое значение конфигурационной переменной `users_par_style` прведет к тому, что в генерируемых CGI-программой **users** html-страницах все абзацы будут помечены следующим образом:

```
<p class="common">A paragraph</p>
```

Обратите внимание на пробел в начале строки значения конфигурационной переменной `users_par_style`. Если бы он отсутствовал, тег `p` и атрибут `class` слились в одно слово `pclass`.

2.10.20 Элемент `users_table_style`

`users_table_style`

Имя элемента: **`users_table_style`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: строка
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данный элемент позволяет задать параметры тегов `<th>` и `<td>` в html-странице, содержащей таблицу всех участников, зарегистрированных на турнир, генерируемой программой `users`. Значение элемента `users_table_style` добавляется в открывающие теги `<th>` и `<td>`. Закрывающие теги `</th>` и `</td>` остаются без изменений.

Пример.

```
<users_table_style> class="reg"</users_table_style>
```

Такое значение конфигурационной переменной `users_table_style` прведет к тому, что в генерируемой CGI-программой **users** таблице участников турнира все заголовки столбцов и ячейки таблицы будут помечены следующим образом:

```
<td class="reg">User</td>
```

Обратите внимание на пробел в начале строки значения конфигурационной переменной `users_table_style`. Если бы он отсутствовал, тег `td` и атрибут `class` слились в одно слово `tdclass`.

2.10.21 Элемент `users_verb_style`

`users_verb_style`

Имя элемента: **`users_verb_style`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: строка
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данный элемент позволяет задать параметры тегов `<th>` и `<td>` в html-странице, содержащей подробную информацию об одном участнике и генерируемой программой `users`. Значение элемента `users_verb_style` добавляется в открывающие теги `<th>` и `<td>`. Закрывающие теги `</th>` и `</td>` остаются без изменений.

Пример.

```
<users_verb_style> class="verb"</users_verb_style>
```

Такое значение конфигурационной переменной `users_verb_style` прведет к тому, что в генерируемой CGI-программой **users** html-странице с подробной информацией об участнике все заголовки столбцов и ячейки таблицы будут помечены следующим образом:

```
<td class="verb">User</td>
```

Обратите внимание на пробел в начале строки значения конфигурационной переменной `users_verb_style`. Если бы он отсутствовал, тег `td` и атрибут `class` слились в одно слово `tdclass`.

2.10.22 Элемент `register_head_style`

`register_head_style`

Имя элемента: **`register_head_style`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: строка
Может отсутствовать: *да*
Значение по умолчанию: `h2`
Может повторяться: *нет*

Описание. Данный элемент позволяет установить тег языка HTML, который используется для выделения названий секций в html-странице, генерируемой CGI-программой register. Если значение этого элемента не установлено, используется тег `h2`, то есть заголовки в генерируемом html-документе будут иметь примерно следующий вид.

```
<h2>Персональная информация пользователя X</h2>
```

Пример.

```
<register_head_style>h3</register_head_style>
```

Такое значение данной конфигурационной переменной приведёт к тому, что тот же самый заголовок будет иметь следующий вид.

```
<h3>Персональная информация пользователя X</h3>
```

Замечание. В текущей версии системы ejudge отсутствует возможность задать отдельно открывающий и закрывающий тег. И для того, и для другого будет использоваться значение данной переменной. Поэтому указание ещё каких-либо дополнительных атрибутов тега заголовка приведёт к тому, что дополнительные атрибуты продублируются в закрывающем теге, приводя, строго говоря, к неправильному html.

2.10.23 Элемент `register_par_style`

`register_par_style`

Имя элемента: **`register_par_style`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: строка
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данный элемент позволяет задать параметры тега в html-страницах, генерируемых программой `register`. Значение элемента `register_par_style` добавляется в открывающий тег . Закрывающий тег остается без изменений.

Пример.

```
<register_par_style> class="common"</register_par_style>
```

Такое значение конфигурационной переменной `register_par_style` прведет к тому, что в генерируемых CGI-программой **register** html-страницах все абзацы будут помечены следующим образом:

```
<p class="common">A paragraph</p>
```

Обратите внимание на пробел в начале строки значения конфигурационной переменной `register_par_style`. Если бы он отсутствовал, тег `p` и атрибут `class` слились в одно слово `pclass`.

2.10.24 Элемент `register_table_style`

`register_table_style`

Имя элемента: **`register_table_style`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: строка
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данный элемент позволяет задать параметры тегов `<th>` и `<td>` в html-странице, содержащей таблицу всех турниров, на которые зарегистрировался данный пользователь, генерируемой программой `register`. Значение элемента `register_table_style` добавляется в открывающие теги `<th>` и `<td>`. Закрывающие теги `</th>` и `</td>` остаются без изменений.

Пример.

```
<register_table_style> class="reg"</register_table_style>
```

Такое значение конфигурационной переменной `register_table_style` прведет к тому, что в генерируемой CGI-программой **register** таблице участников турнира все заголовки столбцов и ячейки таблицы будут помечены следующим образом:

```
<td class="reg">User</td>
```

Обратите внимание на пробел в начале строки значения конфигурационной переменной `register_table_style`. Если бы он отсутствовал, тег `td` и атрибут `class` слились в одно слово `tdclass`.

2.10.25 Элемент `team_head_style`

`team_head_style`

Имя элемента: **`team_head_style`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: строка
Может отсутствовать: *да*
Значение по умолчанию: `h2`
Может повторяться: *нет*

Описание. Данный элемент позволяет установить тег языка HTML, который используется для выделения названий секций в html-странице, генерируемой CGI-программой team. Если значение этого элемента не установлено, используется тег `h2`, то есть заголовки в генерируемом html-документе будут иметь примерно следующий вид.

```
<h2>Персональная страница участника X</h2>
```

Пример.

```
<team_head_style>h3</team_head_style>
```

Такое значение данной конфигурационной переменной приведёт к тому, что тот же самый заголовок будет иметь следующий вид.

```
<h3>Персональная страница участника X</h3>
```

Замечание. В текущей версии системы ejudge отсутствует возможность задать отдельно открывающий и закрывающий тег. И для того, и для другого будет использоваться значение данной переменной. Поэтому указание ещё каких-либо дополнительных атрибутов тега заголовка приведёт к тому, что дополнительные атрибуты продублируются в закрывающем теге, приводя, строго говоря, к неправильному html.

2.10.26 Элемент `team_par_style`

team_par_style

Имя элемента: **`team_par_style`**
Содержится в: contest
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: строка
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данный элемент позволяет задать параметры тега в html-страницах, генерируемых программой team. Значение элемента team_par_style добавляется в открывающий тег . Закрывающий тег остается без изменений.

Пример.

```
<team_par_style> class="common"</team_par_style>
```

Такое значение конфигурационной переменной team_par_style прведет к тому, что в генерируемых CGI-программой team html-страницах все абзацы будут помечены следующим образом:

```
<p class="common">A paragraph</p>
```

Обратите внимание на пробел в начале строки значения конфигурационной переменной team_par_style. Если бы он отсутствовал, тег p и атрибут class слились в одно слово pclass.

2.10.27 Элемент `register_email`

Имя элемента: **`register_email`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: e-mail адрес
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать адрес отправителя (поле `From`) электронного письма, посылаемого пользователю при регистрации в системе. Это письмо посылается по адресу, который был задан пользователем в регистрационной анкете и содержит регистрационное имя, пароль пользователя, а также URL, для дальнейшей регистрации в системе. Если данный элемент в конфигурационном файле турнира не задан, используется значение элемента `register_email` конфигурационного файла `ejudge.xml`. Если и тот элемент не задан, используется адрес по умолчанию `team@contest.cmc.msu.ru`.

Пример.

```
<register_email>register@supercontest.ru</register_email>
```

2.10.28 Элемент `register_url`

Имя элемента: **`register_url`**
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: URL
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать URL CGI-программы register для дальнейшей регистрации в системе ejudge. Этот URL будет помещён в текст письма-уведомления, посылаемого пользователю при регистрации в системе. Если в конфигурационном файле турнира `contest.xml` данный элемент не задан, используется URL, заданный в элементе `register_url` конфигурационного файла `ejudge.xml`. Если и тот элемент не задан, используется URL по умолчанию `http://contest.cmc.msu.ru/cgi-bin/register`. К URL, заданному в любом из конфигурационных файлов или по умолчанию, автоматически добавляется параметр `contest_id` со значением, равным идентификатору данного турнира, параметр `locale_id` со значением, равным идентификатору выбранного языка интерфейса, параметр `login` со значением, равным регистрационному имени нового пользователя, и параметр `action` со значением `3` для перехода непосредственно к странице входа в систему. Таким образом, полностью сформированный URL для примера ниже может выглядеть следующим образом:

```
http://www.supercontest.ru/cgi-bin/register?action=3&login=user&contest_id=4&locale_id=1
```

Пример.

```
<register_url>http://www.supercontest.ru/cgi-bin/register</register_url>
```

2.10.29 Элемент `team_url`

Имя элемента: `team_url`
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: URL
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задать URL для CGI-программы team, которая позволяет пользователю непосредственно участвовать в турнире. Если данный элемент задан, то ссылка на указанный URL будет выведена на личной странице участника после успешной регистрации участника в данном турнире, то есть когда статус регистрации участника установлен в `OK`. Если элемент не задан, ссылка не демонстрируется.

Пример.

```
<team_url>http://www.supercontest.ru/cgi-bin/team</team_url>
```

2.10.30 Элемент `registration_deadline`

Имя элемента: `registration_deadline`
Содержится в: `contest`
Может содержать: *нет*
Атрибуты: *нет*
Тип содержимого: *астрономическое время*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данный элемент позволяет задавать крайний срок приёма регистрационных заявок на данный турнир. Когда астрономическое время станет больше указанного, заявки на участие в данном турнире приниматься не будут, и информация о возможности регистрации на данный турнир не будет выводиться на личной странице участников. Если данный элемент не задан, регистрация не имеет ограничений во времени.

Дата должна задаваться в формате *Year*/*Month*/*Day* *Hour*:*Min*:*Sec*. Здесь *Year* — четырёхзначный номер года от 1906 до 2038 (диапазон представимых дат во внутреннем формате для систем POSIX). *Month* — номер месяца от 1 до 12, *Day* — номер дня в месяце от 1 до 31, *Hour* — час суток от 0 до 23, *Min* — минуты от 0 до 59, *Sec* — секунды от 0 до 59. Элементы даты могут опускаться, начиная с последнего. В этом случае подразумевается минимальное значение. То есть, в спецификации календарного времени могут быть опущены секунды (например, `2003/02/28 14:12`), секунды и минуты (например, `2003/02/28 14`), секунды, минуты и часы (например, `2003/02/28`). Поскольку для конвертации астрономического времени во внутреннее представление используется функция `mktime(3)`, проверка на допустимость даты, например, `2003/02/30`, не производится. Такая дата окажется автоматически преобразованной в `2003/03/02`.

Пример.

```
<registration_deadline>2003/10/05 00:00:00</registration_deadline>
```

`%L`	Заменяется на идентификатор языкового окружения, выбранного пользователем. Английскому языку соответствует идентификатор 0, а русскому — 1.
`%C`	Заменяется на кодировку (charset) страницы, например, `koi8-r`.
`%T`	Заменяется на тип содержимого (content-type) страницы, например, `text/html`.
`%H`	Заменяется на заголовок страницы, который может включать в себя название турнира и/или имя пользователя.
`%%`	Заменяется на один знак `%`.

Таблица 2.6: Специальные конструкции подстановки в файлах-заголовках

`%R`	Заменяется на информацию об авторских правах, правах на распространение (GNU GPL) и версии системы ejudge.
`%%`	Заменяется на один знак `%`.

Таблица 2.7: Специальные конструкции подстановки в файлах-подвалах

2.11 Конфигурационный файл сервера турнира `serve.cfg`

В данной главе описывается формат конфигурационного файла сервера турнира. Он используется программами serve, compile, run.

2.11.1 Общая структура

Конфигурационный файл `serve.cfg` представляет собой текстовый файл в формате, похожем на формат `.ini`-файлов. Комментарии в конфигурационном файле начинаются с символа `#` или `;` и продолжаются до конца строки. Использование символа начала комментария `#` не рекомендуется, если конфигурационный файл предполагается препроцессировать перед чтением (см. ключ `-E` программ serve, compile, run). В этом случае начало комментария, например

```
# include some more information
```

может совпасть с какой-либо директивой препроцессора Си[1]. Кроме того, в конфигурационном файле игнорируются пустые строки.

Конфигурационный файл разбивается на секции. Каждая секция начинается со строки вида

```
[SECTION_NAME]
```

где `SECTION_NAME` — имя секции, например `problem`. В начале конфигурационного файла находится секция глобальных параметров, которая не имеет заголовка.

Каждый конфигурационный параметр задаётся в отдельной строке в одной из следующих форм.

```
NAME = VALUE
```

Перед именем конфигурационной переменной, перед и после знака равенства и после значения параметра может находиться произвольное число пробельных символов. Имя переменной `NAME` состоит из латинских заглавных и строчных букв, цифр, знака подчёркивания. Значение `VALUE` состоит из произвольных непробельных символов. Данная форма задания конфигурационных переменных может использоваться, когда значение переменной не содержит пробельных символов.

```
NAME = "VALUE"
```

Перед именем конфигурационной переменной, вокруг знака равенства и после закрывающей кавычки может находиться произвольное количество пробельных символов. Имя переменной `NAME` состоит из латинских заглавных и строчных букв, цифр, знака подчёркивания. Значение конфигурационной переменной `VALUE` может содержать произвольные символы (включая пробельные) кроме символа двойной кавычки. Данная форма задания конфигурационной переменной может использоваться, если значение переменной содержит пробельные символы, но не содержит кавычек.

```
NAME
```

[1] Судя по всему, использование препроцессора — не очень удачная возможность. Текущая версия системы ejudge позволяет в большинстве случаев не использовать препроцессор.

Перед именем конфигурационной переменной и после него может находиться произвольное количество пробельных символов. Данная форма задания параметра эквивалента форме

```
NAME = 1
```

Конфигурационный файл `serve.cfg` состоит из глобальной секции, в которой задаются значения глобальных конфигурационных параметров, одной или нескольких секций описания задач `problem`, одной или нескольких секций описания поддерживаемых языков программирования `language` и одной или нескольких секций описания процедуры тестирования решений `tester`.

2.11.2 Форматные подстановки

Некоторые конфигурационные переменные определяют не непосредственно значения, используемые при работе сервера турнира, а *шаблоны* значений. Например, конфигурационная переменная `team_info_url` определяет шаблон URL, который используется при генерации строки таблицы результатов для каждой команды. Этот URL содержит ссылку на страницу с подробной информацией о соответствующем участнике. Поскольку URL зависит от некоторых атрибутов участника (например, от его идентификатора), шаблон URL перед использованием должен быть подвергнут преобразованию.

Другой случай, когда выполняется подстановка шаблонов, возникает при наследовании свойств задачи или тестера от абстрактной задачи или тестера соответственно. Например, конфигурационная переменная `input_name` описания задачи задаёт имя файла с входными данными для решения задачи. Это имя файла может быть как фиксированным (например, `input.txt`), так и зависеть от имени задачи (например, `a.in`, `b.in` и т. д.). В последнем случае в описании абстрактной задачи удобно указать шаблон имени входного файла, который автоматически будет раскрыт при наследовании свойств абстрактной задачи конкретной задачей.

В обоих случаях в необходимый момент выполняются *форматные подстановки*, которые описываются ниже. Аргументами форматной подстановки являются: форматная строка, определяющая выполняемые подстановки; информация о текущей задаче (если доступна); информация о текущей команде (если доступна); информация о текущем тестере (если доступна). Результат форматной подстановки записывается в выходную строку.

Форматная строка состоит из символов, копируемых в выходную строку без изменений, и спецификаций форматного преобразования. Спецификация форматного преобразования начинается со знака «процент» (`%`) и имеет следующий вид:

```
%[<флаги>][<ширина>][.<точность>]<спецификатор>
```

Здесь в квадратных скобках записаны необязательные элементы спецификации.

Элемент `<флаги>` позволяет использовать дополнительные возможности преобразования. Элемент `<флаги>` — это последовательность односимвольных флагов, перечисленных в таблице 2.8. Если в одной спецификации форматного преобразования указаны конфликтующие флаги (например, `l` и `u`), действует флаг, указанный последним.

Элемент `<ширина>` задаёт *минимальную* ширину поля. Элемент представляет собой последовательность цифр, формирующих целое число в десятичной записи. Данный элемент работает аналогично элементу, задающему ширину в форматном преобразовании в функциях семейства `printf` языка Си. Если не указаны флаги `r`, `0` или `c`, форматная строка, полученная в результате форматного преобразования, выравнивается по левому краю. Если длина

e	Если форматное преобразование даёт пустую строку, она заменяется на строку . Этот флаг может использоваться для генерации содержимого ячеек html-таблиц.
u	Преобразовать строку, полученную в результате форматного преобразования, к верхнему регистру. Например, если короткое имя (short_name) задач в турнире имеет вид a, b и т. д., а URL, по которому находятся их условия, имеет вид A.html и т. д., то флаг u вместе с преобразованием Ps (то есть спецификация форматного преобразования %uPs) даст необходимый результат.
l	Преобразовать строку, полученную в результате форматного преобразования, к нижнему регистру. Например, если короткое имя задачи (short_name) задач в турнире имеет вид A, B и т. д., а имя входного файла — a.in и т. д., то спецификация форматного преобразования %lPs даст необходимый результат.
r	Выровнять строку, полученную в результате форматного преобразования, по правому краю, дополняя её слева пробелами при необходимости.
0	Выровнять строку, полученную в результате форматного преобразования, по правому краю, дополняя её слева необходимым количеством нулей.
c	Выровнять строку, полученную в результате форматного преобразования, по центру, дополняя её слева и справа пробелами при необходимости.

Таблица 2.8: Флаги спецификации форматного преобразования

строки, полученной в результате форматного преобразования и отсечения по максимальной длине, задаваемого элементом <точность>, меньше указанной в элементе <ширина>, строка дополняется справа необходимым количеством пробелов. Если указан какой-либо из флагов r, 0 или c, и длина строки, полученной в результате форматного преобразования и отсечения по максимальной длине, задаваемого элементом <точность>, меньше указанной в элементе <ширина>, строка выравнивается согласно указанному флагу. Если длина строки, полученной в результате форматного преобразования и отсечения по максимальной длине, задаваемого элементом <точность>, больше указанной в элементе <ширина>, никакого выравнивания не происходит. Если элемент <ширина> не указан, его значение подразумевается равным нулю.

Элемент <точность> задаёт *максимальную* длину строки, получаемой в результате раскрытия спецификатора форматного преобразования. Элемент представляет собой последовательность цифр, формирующих целое число в десятичной записи. Данный элемент работает аналогично элементу, задающему точность в форматном преобразовании в функциях семейства printf языка Си. Раскрытие спецификатора форматного преобразования даёт строку, которая подвергается ограничению на максимальную длину. Например, спецификатор форматного преобразования Ps раскрывается в строку короткого имени (short_name) текущей задачи. Если результат раскрытия спецификатора длиннее, чем указано в элементе <точность>, он обрезается по заданной длине. Если результат раскрытия спецификатора короче, чем указано в элементе <точность>, он сохраняется неизменным. Строка, полученная после применения ограничения на максимальную длину, подвергается преобразованиям выравнивания до минимальной длины, как описано выше. Если элемент <точность> не задан, его значение подразумевается неограниченным.

Элемент <спецификатор> (спецификатор форматного преобразования) указывает данные, которые подставляются вместо спецификации форматного преобразования. Спецификатор состоит из двух букв, где первая буква — это секция, из которой берётся информация: P — текущая задача, T — текущий тестер, M — текущий участник. Вторая буква определяет поле секции. Все поддерживаемые спецификаторы форматного преобразования перечислены в таблице 2.9.

2.11.3 Список глобальных параметров

Название	Стр.	Описание
accept_sound	112	Звуковой файл, проигрываемый при успешной сдаче решения.
archive_dir	92	Каталог для хранения архивов турнира.
auto_short_problem_name	128	Флаг автоматической генерации короткого имени задачи.
autoupdate_standings	115	Флаг автоматического обновления таблицы результатов.
board_fog_time	81	Время заморозки таблицы результатов.
board_unfog_time	81	Время разморозки таблицы результатов.
charset	85	Кодировка генерируемых html-страниц.
checker_dir	91	Каталог для запуска проверяемых программ.
checker_real_time_limit	129	Ограничение времени на работу проверяющей программы.

Продолжение на следующей странице

Название	Стр.	Описание
clar_archive_dir	93	Каталог для хранения сообщений участников и судей.
clar_log_file	105	Файл журнала сообщений участников и судей.
compile_dir	100	Каталог обмена программ serve и compile.
compile_real_time_limit	129	Ограничение времени на компиляцию программ.
compile_work_dir	97	Рабочий каталог программы compile.
conf_dir	88	Каталог конфигурационных файлов.
contest_id	78	Идентификатор турнира.
contests_dir	78	Каталог описания турниров.
contest_time	80	Продолжительность турнира.
corr_dir	90	Каталог с правильными ответами к тестам.
corr_sfx	111	Суффикс файлов с правильными ответами к тестам.
cr_serialization_key	105	Ключ сериализационного семафора.
disable_auto_testing	133	Флаг отключения автоматического тестирования.
disable_clars	125	Флаг полного запрета сообщений.
disable_team_clars	126	Флаг запрета вопросов от участников турнира.
enable_continue	123	Флаг разрешения продолжения турнира.
enable_l10n	86	Флаг включения локализации сообщений.
enable_runlog_merge	133	Флаг включения средств слияния журнала посылок.
ignore_compile_errors	126	Флаг игнорирования ошибок компиляции.
ignore_duplicated_runs	127	Флаг игнорирования дублирующих посылок.
inactivity_timeout	106	Тайм-аут программы serve.
info_dir	135	Каталог с дополнительной информацией о тестах.
info_sfx	137	Суффикс файлов с дополнительной информацией о тестах.
internal_sound	114	Звук, проигрываемый при внутренней ошибке тестирования.
l10n_dir	86	Каталог файлов локализации.
max_clar_num	84	Квота вопросов от одной команды.
max_clar_size	83	Максимальный размер одного вопроса от команды.
max_clar_total	84	Квота суммарного размера вопросов от одной команды.
max_cmd_length	134	Максимальный размер включаемых в протокол тестирования аргументов командной строки.
max_file_length	128	Максимальная длина включаемого в протокол тестирования файла.
max_line_length	127	Максимальная длина строки в протоколе тестирования
max_run_num	83	Квота количества решений от одной команды.
max_run_size	82	Максимальный размер одного решения от команды.
max_run_total	82	Квота суммарного размера решений от одной команды.
plog_file_name	118	Имя файла с публичным журналом посылок.
plog_footer_file	120	Файл-подвал публичного журнала посылок.
plog_header_file	119	Файл-заголовок публичного журнала посылок.

Продолжение на следующей странице

Название	Стр.	Описание
plog_update_time	119	Период обновления публичного журнала посылок.
presentation_sound	114	Звук, проигрываемый при presentation error.
prob_info_url	121	Шаблон URL текста задачи.
prune_empty_users	138	Флаг игнорирования «пустых» пользователей.
report_archive_dir	94	Каталог архива протоколов тестирования.
report_error_code	123	Флаг разрешения печати кода возврата в протокол участника.
root_dir	87	Корневой каталог файлов турнира.
run_archive_dir	93	Каталог архива посылок.
run_check_dir	99	Рабочий каталог для проверки решений.
run_dir	102	Каталог обмена программ **serve** и **run**.
run_log_file	104	Файл журнала посылок.
runtime_sound	112	Звук, проигрываемый при runtime error.
run_work_dir	98	Рабочий каталог программы **run**.
score_system	107	Тип турнира.
script_dir	88	Каталог скриптов компиляции и запуска.
serve_sleep_time	80	Период опроса каталогов обмена программой **serve**.
serve_socket	104	UNIX-сокет команд программе **serve**.
show_astr_time	124	Флаг астрономического времени при печати журнала посылок.
show_deadline	130	Флаг печати крайнего времени сдачи в меню задач.
sleep_time	79	Период опроса каталогов обмена.
socket_path	79	UNIX-сокет для связи с программой **userlist-server**.
sound_player	111	Программа, которая проигрывает звуковые файлы.
stand2_file_name	117	Имя вторичного файла с таблицей текущих результатов.
stand2_footer_file	118	Файл-подвал вторичного файла таблицы текущих результатов.
stand2_header_file	117	Файл-заголовок вторичного файла таблицы текущих результатов.
standings_charset	85	Кодировка файла таблицы результатов.
standings_file_name	115	Файл основной таблицы результатов.
standings_locale	87	Язык файла таблицы результатов.
stand_extra_attr	143	Атрибуты дополнительной пользовательской колонки таблицы результатов.
stand_extra_format	143	Формат вывода в дополнительной пользовательской колонке таблицы результатов.
stand_extra_legend	143	Заголовок дополнительной пользовательской колонки таблицы результатов.
stand_footer_file	116	Файл-подвал основного файла таблицы текущих результатов.
stand_header_file	116	Файл-заголовок основного файла таблицы текущих результатов.
stand_penalty_attr	142	Атрибуты колонки «штраф» таблицы текущих результатов (в режиме турнира *ACM*).

Продолжение на следующей странице

Название	Стр.	Описание
stand_place_attr	139	Атрибуты колонки «место» таблицы текущих результатов.
stand_prob_attr	140	Атрибуты колонок «задача» таблицы текущих результатов.
stand_r_row_attr	145	Атрибуты строки таблицы текущих результатов для невиртуальных участников виртуальных турниров и для участников невиртуальных турниров.
stand_score_attr	141	Атрибуты колонки «очки» таблицы текущих результатов (в режимах турнира *kirov* или *olympiad*).
stand_self_row_attr	144	Атрибуты строки таблицы текущих результатов, соответствующей текущей команде при выводе текущих результатов данной команды в виртуальном турнире (только для виртуальных турниров).
stand_solved_attr	141	Атрибуты колонки «количество решенных задач» таблицы текущих результатов.
stand_table_attr	139	Атрибуты тега <table> таблицы текущих результатов.
stand_team_attr	140	Атрибуты колонки «команда» таблицы текущих результатов.
stand_time_attr	142	Атрибуты для вывода затраченного времени в ячейке таблицы текущих результатов (в режиме турнира ACM).
stand_u_row_attr	145	Атрибуты строки таблицы текущих результатов для участников виртуального турнира с неопределенным статусом.
stand_v_row_attr	144	Атрибуты строки таблицы текущих результатов для виртуальных участников виртуальных турниров.
status_dir	95	Каталог файлов состояния программы **serve**.
team_download_time	124	Период скачивания участником своих решений.
team_enable_rep_view	122	Флаг разрешения просмотра участником протоколов тестирования.
team_enable_src_view	122	Флаг разрешения просмотра участником исходного текста решений.
team_info_url	121	Шаблон URL с информацией об участнике.
team_report_archive_dir	94	Каталог архива протоколов тестирования, предназначенных для участников.
test_dir	89	Каталог с тестами к задачам.
test_sfx	110	Суффикс имён файлов с тестами.
tests_to_accept	125	Количество тестов для принятия задачи в режиме *OLYMPIAD*.
tgz_dir	136	Каталог с архивами рабочих каталогов для тестирования решений.
tgz_sfx	137	Суффиксы архивов рабочих каталогов.
timelimit_sound	113	Звук, проигрываемый при истечении лимита времени.
variant_map_file	131	Путь к файлу с распределением вариантов между участниками.

Продолжение на следующей странице

Название	Стр.	Описание
var_dir	92	Каталог рабочих файлов турнира.
virtual	109	Флаг виртуального турнира.
work_dir	96	Рабочий каталог.
wrong_sound	113	Звук, проигрываемый при неправильном ответе тестируемой программы.

Текущая задача (секция problem)	
Pi	Идентификатор задачи (поле id)
Ps	Короткое имя задачи (поле short_name)
Pl	Полное имя задачи (поле name)
Текущий тестер (секция tester)	
Ti	Идентификатор тестера (поле id)
Tn	Имя тестера (поле name)
Tj	Идентификатор задачи, которую проверяет тестер (поле problem)
Tp	Короткое имя задачи, которую проверяет тестер (поле problem_name)
Ta	Архитектура, на которой работает тестер (поле arch)
Tk	Ключ тестера (поле key)
Текущий участник	
Mi	Идентификатор участника
Mn	Имя участника
Ml	Регистрационное имя (login) участника
Mc	Поле «город» (city) регистрационной формы участника. Если данное поле не заполнялось, заменяется на пустую строку
MC	Поле «город (англ.)» (city_en) регистрационной формы участника
Mo	Поле «страна» (country) регистрационной формы участника
MO	Поле «страна (англ.)» (country_en) регистрационной формы участника
Mt	Поле «краткое название учебного заведения» (inst_short) регистрационной формы участника
MT	Поле «краткое название учебного заведения (англ.)» (inst_short_en) регистрационной формы участника
Mu	Поле «название учебного заведения» (inst) регистрационной формы участника
MU	Поле «название учебного заведения (англ.)» (inst_en) регистрационной формы участника

Таблица 2.9: Спецификаторы форматного преобразования

2.11.4 Глобальные параметры

В данном разделе рассматриваются глобальные конфигурационные переменные конфигурационного файла `serve.cfg`.

Переменная `contest_id`

Имя переменной: **`contest_id`**
Содержится в: `global`
Используется: serve
Тип содержимого: *integer*
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт идентификатор обслуживаемого турнира. Идентификатор турнира — целое число больше нуля. Название турнира и информация общего характера о турнире берётся из конфигурационного файла турнира `contest.xml`.

Пример.

```
contest_id = 20
```

Переменная `contests_dir`

Имя переменной: **`contests_dir`**
Содержится в: `global`
Используется: serve
Тип содержимого: *путь к каталогу*
Может отсутствовать: *нет*
Может повторяться: *нет*

Данная конфигурационная переменная задаёт путь к каталогу, в котором находятся файлы описания турниров `contest.xml`. Каждое описание турнира находится в отдельном файле с именем, равным шестизначному номеру турнира, например, `000000.xml`. Используемое описание турнира определяется с помощью конфигурационной переменной `contest_id`.

Пример.

```
contests_dir = "/var/ejudge/contests"
```

Переменная `socket_path`

Имя переменной: `socket_path`
Содержится в: `global`
Используется: serve
Тип содержимого: *путь к UNIX-сокету*
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт путь к UNIX-сокету, который используется для связи с сервером информации о пользователях userlist-server. Когда программа serve стартует, UNIX-сокет должен существовать и быть доступным для пользователя, от имени которого запускается программа.

Пример.

```
socket_path = "/var/ejudge/userlist-socket"
```

Переменная `sleep_time`

Имя переменной: `sleep_time`
Содержится в: `global`
Используется: serve, compile, run
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 1000
Может повторяться: *нет*

Программа serve обменивается с программами compile и run посредством разделяемых файлов в общей файловой системе. Например, чтобы отправить полученное от участника решение на компиляцию, программа serve записывает служебный файл и файл решения в специальные каталоги обмена с программой compile. Все три программы периодически проверяют, не появились ли новые файлы в каталогах обмена, и в случае появления новых файлов выполняют соответствующие действия. Переменная `sleep_time` позволяет задать интервал времени в миллисекундах между опросами каталогов обмена для программ serve, compile и run. Для программы serve интервал времени может переопределяться с помощью переменной `serve_sleep_time`. Значение конфигурационной переменной `sleep_time` по умолчанию равно 1000 (миллисекунд), что соответствует задержке в 1 секунду.

Переменная `serve_sleep_time`

Имя переменной: **`serve_sleep_time`**
Содержится в: `global`
Используется: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: равно значению `sleep_time`
Может повторяться: *нет*

Данная переменная задаёт интервал времени между опросами каталогов обмена программы serve. Интервал времени задаётся в миллисекундах. Если конфигурационная переменная `serve_sleep_time` не установлена, используется значение конфигурационной переменной `sleep_time`, а если и эта переменная не установлена, используется значение по умолчанию 1000, что соответствует одной секунде задержки.

Переменная `contest_time`

Имя переменной: **`contest_time`**
Содержится в: `global`
Используется: serve
Тип содержимого: *integer*
Может отсутствовать: *нет*
Может повторяться: *нет*

Данная конфигурационная переменная задаёт время турнира в секундах. Время в секундах должно быть целым значением, больше или равно 0. Если время равно 0, продолжительность турнира **неограничена**. В этом случае параметры `board_fog_time`, `board_unfog_time` не используются. Турнир продолжается до тех пор, пока не будет остановлен администратором.

Пример.

```
contest_time = 0
```

Здесь задаётся неограниченная продолжительность турнира. Обратите внимание, что в этом случае следует изменить значения конфигурационных переменных **TODO** — каких?

```
contest_time = 300
```

В данном случае продолжительность турнира устанавливается равной 5 часам.

Переменная board_fog_time

Имя переменной: **board_fog_time**
Содержится в: global
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 60
Может повторяться: *нет*

По регламенту проведения командных соревнований по программированию ACM таблица текущих результатов «замораживается», то есть перестаёт обновляться за 60 минут до конца соревнования. С помощью переменной board_fog_time можно установить любое другое время заморозки таблицы результатов. Время задаётся в минутах, то есть, например, значение 60 означает заморозку таблицы результатов за час до конца. Значение переменной должно быть больше нуля. Значение 0 эквивалентно значению по умолчанию (60 минут)[2]. Чтобы свести заморозку к минимуму, задайте значение переменной 1. Заморозка действует только на автоматические обновления таблицы результатов. Ручное обновление таблицы возможно вне зависимости от времени заморозки (с помощью CGI-программы master). Если после того, как заморозка вступила в действие, турнир был продлён так, что согласно новой продолжительности турнира заморозка в действие не вступила, таблица результатов будет обновлена, как только поступит новое решение или по команде обновления от администратора турнира.

Пример. Приведённый ниже пример включает заморозку результатов за 30 минут до конца турнира.

```
board_fog_time = 30
```

Переменная board_unfog_time

Имя переменной: **board_unfog_time**
Содержится в: global
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 120
Может повторяться: *нет*

Данная конфигурационная переменная задаёт время в минутах после окончания турнира, после которого снова возможны автоматические обновления таблицы результатов. Ручное обновление таблицы результатов после окончания турнира возможно вне зависимости от значения этой конфигурационной переменной. Значение переменной должно быть больше или равно 0. Если указано значение 0, автоматическое обновление результатов возможно сразу же после окончания турнира.

Пример. Приведённый ниже пример отключает заморозку результатов после окончания турнира.

```
board_unfog_time = 0
```

[2] Это, по-видимости, нужно исправить. 0 должно означать отсутствие заморозки.

Переменная `max_run_size`

Имя переменной: **`max_run_size`**
Содержится в: `global`
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 65535
Может повторяться: *нет*

Данная конфигурационная переменная устанавливает максимальный размер программы, принимаемой на проверку. Если размер программы превышает лимит, программа к проверке не принимается, а участнику выводится сообщение об ошибке. При увеличении максимального размера программы учтите, что действует ограничение размера данных, принимаемых от клиента, равное 128 килобайт. Чтобы изменить это ограничение исправьте значение константы `MAX_VALUE_SIZE` в исходном файле `cgi.c` и перекомпилируйте систему ejudge.

Пример. Данный пример устанавливает максимальный размер программы равным 100 килобайт.

```
max_run_size = 102400
```

Переменная `max_run_total`

Имя переменной: **`max_run_total`**
Содержится в: `global`
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 2097152
Может повторяться: *нет*

Данная конфигурационная переменная устанавливает максимальный суммарный размер всех решений от одного участника (квота размера). Если размер очередной присланной на проверку программы таков, что в сумме с предыдущими программами того же участника он превышает квоту, программа на проверку не принимается, а участнику выдаётся сообщение об ошибке. Размер квоты по умолчанию — 2 мегабайта, чего более чем достаточно для обычного использования.

Переменная `max_run_num`

Имя переменной: **`max_run_num`**
Содержится в: `global`
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 200
Может повторяться: *нет*

Данная конфигурационная переменная устанавливает максимальное количество присылаемых на проверку программ от одного участника. Количество присланных программ суммируется для каждого участника по всем задачам. Если максимальное количество присылаемых программ превышено, очередная программа будет отвергнута, а участнику будет выдано сообщение об ошибке. Значение по умолчанию — 200, чего должно быть достаточно для разумного использования.

Переменная `max_clar_size`

Имя переменной: **`max_clar_size`**
Содержится в: `global`
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 1024
Может повторяться: *нет*

Данная конфигурационная переменная устанавливает максимальный размер сообщения или вопроса от участника. Если размер текста вопроса превышает лимит, вопрос не принимается, а участнику выводится сообщение об ошибке. При увеличении максимального размера сообщения учтите, что действует ограничение размера данных, принимаемых от клиента, равное 128 килобайт. Чтобы изменить это ограничение исправьте значение константы `MAX_VALUE_SIZE` в исходном файле `cgi.c` и перекомпилируйте систему ejudge.

Пример. Данный пример устанавливает максимальный размер сообщения равным 2 килобайта.

```
max_clar_size = 2048
```

Переменная `max_clar_total`

Имя переменной: **`max_clar_total`**
Содержится в: `global`
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 40960
Может повторяться: *нет*

Данная конфигурационная переменная устанавливает максимальный суммарный размер текста всех сообщений от одного участника (квота размера). Если размер очередного сообщения таков, что в сумме с предыдущими сообщениями того же участника он превышает квоту, сообщение не принимается, а участнику выдаётся сообщение об ошибке. Размер квоты по умолчанию — 40 килобайт, чего более чем достаточно для обычного использования.

Переменная `max_clar_num`

Имя переменной: **`max_clar_num`**
Содержится в: `global`
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 50
Может повторяться: *нет*

Данная конфигурационная переменная устанавливает максимальное количество сообщений от одного участника. Если максимальное количество сообщений превышено, очередное сообщение будет отвергнуто, а участнику будет выдано сообщение об ошибке. Значение по умолчанию — 50, чего должно быть достаточно для разумного использования.

Переменная `charset`

Имя переменной: `charset`
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `iso8859-1`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт кодировку, в которой генерируются все html-страницы, передаваемые клиентам team, master, judge. В этой кодировке также сохраняются базы дампы баз данных.

Внимание. В текущей версии системы ejudge поддержка разных кодировок реализована неудовлетворительно. Не гарантируется корректная работа системы, если кодировки, заданные в параметрах `charset` конфигурационных файлов CGI-программ team, master, judge, и кодировка, заданная в этом файле, не совпадают. Кроме того, в действительности поддерживаются только две кодировки: `iso8859-1` и `koi8-r`. Корректная работа с другими кодировками также не гарантируется.

Пример.

```
charset = koi8-r
```

Переменная `standings_charset`

Имя переменной: `standings_charset`
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: совпадает со значением `charset`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт кодировку, в которой генерируется статическая таблица результатов турнира. Статическая таблица результатов — это html-файл, генерируемый в каталоге файлов состояния сервера турнира, который обновляется после каждой посылки участника. Назначение этого файла — дать возможность всем желающим просматривать текущие результаты турнира. По умолчанию предполагается, что кодировка этого файла совпадает с глобальной кодировкой сервера турнира (см. конфигурационную переменную charset). Если переменная `standings_charset` задаёт другую кодировку, выполняется перекодирование файла.

Внимание. Корректная работа системы ejudge в случае, если переменная `standings_charset` задаёт кодировку, отличную от `cp1251`, не гарантируется.

Пример.

```
charset = cp1251
```

Переменная `enable_l10n`

Имя переменной: `enable_l10n`
Содержится в: `global`
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная включает локализацию сообщений во фрагментах html-файлов, генерируемых программой serve для CGI-программ team, judge, master. Если локализация включена, требуемый язык сообщений определяется клиентом. Чтобы включить локализацию, достаточно написать в конфигурационном файле имя переменной `enable_l10n`.

Замечание. В настоящее время только CGI-программа team поддерживает локализацию сообщений. CGI-программы judge и master выводят сообщения только на английском языке. В настоящее время поддерживаются два языка: английский и русский (в кодировке koi8-r). Поскольку для локализации используются стандартные средства локализации (функция `gettext(3)`, доступная на Linux), добавление новых языков является достаточно простой задачей, если существующие проблемы с кодировками будут решены.

Переменная `l10n_dir`

Имя переменной: `l10n_dir`
Содержится в: `global`
Используются: serve
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: (пустая строка)
Может повторяться: *нет*

Описание. Переводы сообщений находятся не в исполняемом файле программы, а в отдельном каталоге. Чтобы сервер турнира serve мог использовать их, необходимо указать полный путь к каталогу сообщений (message catalog), в котором находятся файлы с переведёнными сообщениями. По умолчанию после выполнения команды `make install` каталоги сообщений помещаются в каталог, определяемой переменной `INST_LOCALE_PATH` в `makefile`. Если каталог не указан, или указан неправильно, локализация сообщений работать не будет, даже если параметр `enable_l10n` установлен в *true*.

Пример.

```
enable_l10n
l10n_dir = "/usr/share/locale"
```

Переменная `standings_locale`

Имя переменной: `standings_locale`
Содержится в: global
Используются: serve
Тип содержимого: *строка*
Может отсутствовать: *да*
Значение по умолчанию: (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет локализовать статическую таблицу результатов турнира. По умолчанию все элементы таблицы выводятся на английском языке (например, “Place” в заголовке столбца таблицы). Установка значения данной конфигурационной переменной в имя другого локального окружения (locale) позволяет изменять язык. Например, если включено русское локальное окружение `ru_RU.KOI8-R`, тот же заголовок столбца таблицы будет называться «`Место`». Перекодировка таблицы, управляемая конфигурационной переменной `standings_charset`, выполняется после генерации таблицы результатов.

Внимание. В настоящее время поддерживается только русское локальное окружение `ru_RU.KOI8-R`.

Пример.

```
standings_locale = "ru_RU.KOI8-R"
```

Переменная `root_dir`

Имя переменной: `root_dir`
Содержится в: global
Используются: serve, compile, run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет задать путь к каталогу, в котором находятся конфигурационные и рабочие файлы системы ejudge. В подкаталогах этого каталога рекомендуется размещать конфигурационные файлы serve, тесты, правильные ответы, проверяющие программы. Кроме того, в подкаталоге `var` этого каталога будут размещаться рабочие файлы турнира: архив посылок участников и их вопросов, архив протоколов запуска и временные файлы. Переменная должна быть обязательно задана в конфигурационном файле.

Пример.

```
root_dir = /home/judges/20
```

Переменная `conf_dir`

Имя переменной: **`conf_dir`**
Содержится в: `global`
Используются: serve, compile, run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `conf`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет задать каталог, в котором находятся конфигурационные файлы, тесты, проверяющие программы, ответы к тестам. Если эта переменная не задана, конфигурационным каталогом является подкаталог `conf` каталога, заданного в переменной `root_dir`. Если конфигурационная переменная `conf_dir` задана, полный путь к каталогу конфигурационных файлов образуется конкатенацией пути, указанного в переменной `root_dir` и пути, указанного в переменной `conf_dir`.

Переменная `script_dir`

Имя переменной: **`script_dir`**
Содержится в: `global`
Используются: compile, run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `scripts`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная определяет путь к каталогу скриптов компиляции и запуска программ. В этом каталоге находятся программы (обычно это скрипты на языке bash), вызываемые для компиляции посылок участников, а также вспомогательные программы, назначение которых — запустить проверяемое решение участника. Полный путь к каталогу скриптов определяется по следующим правилам:

- Если значение переменной не задано, используется значение `scripts`.
- Если значение переменной начинается с символа `'/'`, то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной не начинается с символа `'/'`, полный путь к каталогу скриптов образуется конкатенацией значения переменной `conf_dir` и значения переменной `script_dir`.

Пример.

```
script_dir = /usr/share/ejudge/scripts
```

Переменная test_dir

Имя переменной: **test_dir**
Содержится в: global
Используются: run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: tests
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором хранятся тесты для всех задач данного турнира. Сами тесты находятся в подкаталогах этого каталога. Тесты для некоторой задачи с кратким именем A (задаваемом конфигурационной переменной short_name) находятся в подкаталоге A тестового каталога. Полный путь к каталогу с тестами определяется по следующим правилам:

- Если значение переменной test_dir не задано, используется значение tests.
- Если значение переменной test_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной test_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной conf_dir и значения переменной test_dir.

Пример. В следующем примере путь к каталогу тестов устанавливается в ${root_dir}/conf/../tests, где ${root_dir} — значение конфигурационной переменной root_dir. Этот путь эквивалентен пути ${root_dir}/tests.

```
script_dir = ../tests
```

Переменная `corr_dir`

Имя переменной: **`corr_dir`**
Содержится в: `global`
Используются: run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `correct`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором хранятся правильные ответы к тестам для всех задач данного турнира. Сами правильные ответы находятся в подкаталогах этого каталога. Правильные ответы для некоторой задачи с кратким именем `A` (задаваемом конфигурационной переменной `short_name`) находятся в подкаталоге `A` каталога правильных ответов. Полный путь к каталогу с правильными ответами определяется по следующим правилам:

- Если значение переменной `corr_dir` не задано, используется значение `correct`.
- Если значение переменной `corr_dir` начинается с символа `'/'`, то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной `corr_dir` не начинается с символа `'/'`, полный путь к каталогу скриптов образуется конкатенацией значения переменной `conf_dir` и значения переменной `corr_dir`.

Пример. В следующем примере путь к каталогу тестов устанавливается в `${root_dir}/conf/../tests`, где `${root_dir}` — значение конфигурационной переменной `root_dir`. Этот путь эквивалентен пути `${root_dir}/tests`. Таким образом, тесты к задачам и правильные ответы к тестам находятся в одном каталоге.

```
script_dir = ../tests
```

Переменная checker_dir

Имя переменной: **checker_dir**
Содержится в: global
Используются: run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: checkers
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором хранятся проверяющие программы для всех задач данного турнира. Полный путь к проверяющей программе является конкатенацией имени каталога, определённого в данной конфигурационной переменной, и имени самой программы, определённого в конфигурационной переменной check_cmd. Полный путь к каталогу с проверяющими программами определяется по следующим правилам:

- Если значение переменной checker_dir не задано, используется значение checkers.
- Если значение переменной checker_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной checker_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной conf_dir и значения переменной checker_dir.

Пример. В следующем примере путь к каталогу тестов устанавливается в ${root_dir}/conf/../checkers, где ${root_dir} — значение конфигурационной переменной root_dir. Этот путь эквивалентен пути ${root_dir}/checkers.

```
script_dir = ../checkers
```

Переменная `var_dir`

Имя переменной: **`var_dir`**
Содержится в: `global`
Используются: run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `var`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором хранятся архивные и рабочие файлы турнира. Если переменная в конфигурационном файле не установлена, её значение полагается равным `var`. Полный путь к каталогу получается конкатенацией значения переменной `root_dir` и данной переменной.

Внимание. Изменять значение по умолчанию данной переменной не рекомендуется. Изменение должно быть согласовано с конфигурационными файлами CGI-программ.

Переменная `archive_dir`

Имя переменной: **`archive_dir`**
Содержится в: `global`
Используются: serve
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `archive`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором размещаются архивы турнира. Полный путь к каталогу с проверяющими программами определяется по следующим правилам:

- Если значение переменной `archive_dir` не задано, используется значение `archive`.
- Если значение переменной `archive_dir` начинается с символа `'/'`, то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной `archive_dir` не начинается с символа `'/'`, полный путь к каталогу скриптов образуется конкатенацией значения переменной `var_dir` и значения переменной `archive_dir`.

Переменная clar_archive_dir

Имя переменной: **clar_archive_dir**
Содержится в: global
Используются: serve
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: clars
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором размещается архив сообщений турнира. Полный путь к каталогу с архивом сообщений определяется по следующим правилам:

- Если значение переменной clar_archive_dir не задано, используется значение clars.
- Если значение переменной clar_archive_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной clar_archive_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной archive_dir и значения переменной clar_archive_dir.

Переменная run_archive_dir

Имя переменной: **run_archive_dir**
Содержится в: global
Используются: serve
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: runs
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором размещается архив посылок от участников турнира. Полный путь к каталогу с архивом посылок определяется по следующим правилам:

- Если значение переменной run_archive_dir не задано, используется значение runs.
- Если значение переменной run_archive_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной run_archive_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной archive_dir и значения переменной run_archive_dir.

Переменная `report_archive_dir`

Имя переменной: **`report_archive_dir`**
Содержится в: `global`
Используются: serve
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `reports`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором размещается архив протоколов запусков решений участников турнира. Полный путь к каталогу с архивом протоколов определяется по следующим правилам:

- Если значение переменной `report_archive_dir` не задано, используется значение `reports`.
- Если значение переменной `report_archive_dir` начинается с символа `'/'`, то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной `report_archive_dir` не начинается с символа `'/'`, полный путь к каталогу скриптов образуется конкатенацией значения переменной `archive_dir` и значения переменной `report_archive_dir`.

Переменная `team_report_archive_dir`

Имя переменной: **`team_report_archive_dir`**
Содержится в: `global`
Используются: serve
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `teamreports`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором размещается архив протоколов участников запусков решений участников турнира. Полный путь к каталогу с архивом протоколов определяется по следующим правилам:

- Если значение переменной `team_report_archive_dir` не задано, используется значение `teamreports`.
- Если значение переменной `team_report_archive_dir` начинается с символа `'/'`, то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной `team_report_archive_dir` не начинается с символа `'/'`, полный путь к каталогу скриптов образуется конкатенацией значения переменной `archive_dir` и значения переменной `team_report_archive_dir`.

Переменная `status_dir`

Имя переменной: **`status_dir`**
Содержится в: `global`
Используются: serve
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `status`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором находятся файлы состояния турнира. К таким файлам относятся таблица текущих результатов и файл статуса сервера. Полный путь к этому каталогу определяется по следующим правилам:

- Если значение переменной `status_dir` не задано, используется значение `status`.
- Если значение переменной `status_dir` начинается с символа `'/'`, то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной `status_dir` не начинается с символа `'/'`, полный путь к каталогу скриптов образуется конкатенацией значения переменной `var_dir` и значения переменной `status_dir`.

Внимание. Изменять значение по умолчанию данной переменной не рекомендуется. Изменение должно быть согласовано с конфигурационными файлами CGI-программ.

Переменная work_dir

Имя переменной: **work_dir**
Содержится в: global
Используются: compile, run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: work
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, используемому для временных файлов программами compile и run. Каждая из этих программ использует свой подкаталог в этом каталоге. Например, программа compile копирует в рабочий каталог файл исходного текста, компилирует его в исполняемый файл в рабочем каталоге, который затем копируется в каталоги обмена с программой serve. Полный путь к этому каталогу определяется по следующим правилам:

- Если значение переменной work_dir не задано, используется значение work.
- Если значение переменной work_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной work_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной var_dir и значения переменной work_dir.

Внимание. Если планируется параллельная работа нескольких программ compile или нескольких программ run на нескольких компьютерах с обменом через каталоги, монтируемые с сервера, один и тот же рабочий каталог компиляции и запуска программ не должен разделяться несколькими работающими программами.

Переменная compile_work_dir

Имя переменной: **compile_work_dir**
Содержится в: global
Используются: compile, run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: compile
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, используемому для временных файлов программой compile. Программа копирует в рабочий каталог файл исходного текста, компилирует его в исполняемый файл в рабочем каталоге, который затем копируется в каталоги обмена с программой serve. Полный путь к этому каталогу определяется по следующим правилам:

- Если значение переменной compile_work_dir не задано, используется значение compile.
- Если значение переменной compile_work_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной compile_work_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной work_dir и значения переменной compile_work_dir.

Внимание. Если планируется параллельная работа нескольких программ compile на нескольких компьютерах с обменом через каталоги, монтируемые с сервера, один и тот же рабочий каталог компиляции программ не должен разделяться несколькими работающими программами. Наилучший способ добиться этого — расположить рабочий каталог на локальном диске.

Пример.

```
compile_work_dir = /tmp/compile
```

Переменная run_work_dir

Имя переменной: **run_work_dir**
Содержится в: global
Используются: compile, run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: runwork
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, используемому для временных файлов программой run. В этом каталоге хранится временная копия тестируемой программы, а также файлы протокола запуска тестируемой программы. Тестируемые программы запускаются в другом каталоге, задаваемом конфигурационной переменной run_check_dir. После завершения проверки программы рабочий каталог очищается. Полный путь к этому каталогу определяется по следующим правилам:

- Если значение переменной run_work_dir не задано, используется значение runwork.
- Если значение переменной run_work_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной run_work_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной work_dir и значения переменной run_work_dir.

Внимание. Если планируется параллельная работа нескольких программ run на нескольких компьютерах с обменом через каталоги, монтируемые с сервера, один и тот же рабочий каталог запуска программ не должен разделяться несколькими работающими программами. Наилучший способ добиться этого — расположить рабочий каталог на локальном диске.

Пример.

```
run_work_dir = /tmp/runwork
```

Переменная run_check_dir

Имя переменной: **run_check_dir**
Содержится в: global
Используются: run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: runcheck
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, используемому для тестирования программой run. Для запуска проверяемой программы на каждом тесте в этот каталог копируется сама программа, затем входной файл теста. В каталоге создаются пустые выходные файлы. После этого запускается тестируемая программа. После каждого запуска и проверки результатов содержимое каталога очищается. Полный путь к этому каталогу определяется по следующим правилам:

- Если значение переменной run_check_dir не задано, используется значение runcheck.
- Если значение переменной run_check_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной run_check_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной work_dir и значения переменной run_check_dir.

Внимание. Если планируется параллельная работа нескольких программ run на нескольких компьютерах с обменом через каталоги, монтируемые с сервера, один и тот же каталог запуска программ не должен разделяться несколькими работающими программами. Наилучший способ добиться этого — расположить рабочий каталог на локальном диске.

Пример.

```
run_check_dir = /tmp/runcheck
```

Переменная `compile_dir`

Имя переменной: **`compile_dir`**
Содержится в: `global`
Используются: serve, compile
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `compile`
Может повторяться: *нет*

Описание. Данная переменная задаёт каталоги обмена между программами serve и compile. Полный путь к этому каталогу определяется по следующим правилам:

- Если значение переменной `compile_dir` не задано, используется значение `compile`.
- Если значение переменной `compile_dir` начинается с символа `'/'`, то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной `compile_dir` не начинается с символа `'/'`, полный путь к каталогу скриптов образуется конкатенацией значения переменной `var_dir` и значения переменной `compile_dir`.

Каталог обмена содержит в себе несколько подкаталогов, назначение которых описывается ниже.

- Подкаталог **`queue`** используется для передачи информации от программы serve к программе compile. Программа serve помещает в этот каталог небольшие файлы, содержащие служебную информацию о программе, которую необходимо скомпилировать (так называемый «пакет задания компиляции»). Программа compile периодически просматривает этот каталог и при появлении в нём новых пакетов задания компиляции выполняет их. Этот каталог может использоваться совместно несколькими программами serve, обслуживающими разные турниры, и несколькими программами compile, например, работающими на разных компьютерах в сети. Этот каталог имеет специальную структуру, чтобы предотвратить синхронизационные ошибки при одновременном доступе нескольких программ.
- Подкаталог **`src`** используется для передачи информации от программы serve к программе compile. Программа serve помещает в этот каталог файл исходного текста программы, которую необходимо скомпилировать. Файл с текстом программы имеет то же самое имя, что и пакет задания компиляции для этой программы. Программа compile считывает файл исходного текста после файла пакета задания компиляции. Этот каталог может использоваться совместно несколькими программами serve, обслуживающими разные турниры, и несколькими программами compile. Синхронизация доступа поддерживается в каталоге `queue`, поэтому никакой дополнительной синхронизации в каталоге `src` не требуется.
- Подкаталог **`status`** используется для передачи информации от программы compile к программе serve. В этот каталог программа compile помещает небольшой файл с информацией о результате компиляции (так называемый «пакет результата компиляции»). Программа serve периодически просматривает данный каталог и при появлении в нём

новых файлов обновляет своё внутреннее состояние. Поскольку каталог может использоваться одновременно несколькими программами compile и одной программой serve, он имеет специальную структуру для предотвращения синхронизационных ошибок.

- Подкаталог **`report`** используется для передачи информации от программы compile к программе serve. В этот каталог программа compile помещает список ошибок, выданных при неуспешной компиляции, или скомпилированный исполняемый файл в случае успешной компиляции. Каталог может использоваться одновременно несколькими программами compile и одной программой serve, но синхронизация доступа ведётся с помощью каталога `status`, поэтому никакой дополнительной синхронизации в каталоге `report` не требуется.

- Символическая ссылка **`<номер>`**, где `<номер>` — четырёхзначный идентификатор турнира. Эта символическая ссылка создаётся в каталоге, задаваемом переменной `compile_dir`, программы compile, и указывает на каталог, задаваемый переменной `compile_dir`, программы serve. Если эти каталоги совпадают, символическая ссылка указывает на каталог, в котором она находится. С помощью этой символической ссылки программа compile может обслуживать одновременно несколько турниров, записывая результаты компиляции в каталог обмена только соответствующего турнира.

Пример. С помощью задания каталога обмена можно добиться того, что несколько одновременно работающих серверов турнира будут использовать одну программу компиляции compile. Для этого создаётся отдельный каталог для программы compile, например `/var/ejudge/compile`. Этот каталог указывается в качестве коревого в конфигурационном файле программы compile с помощью строки

```
root_dir = /var/ejudge/compile
```

Программа compile запускается в каталоге `/var/ejudge/compile`. Теперь во всех конфигурационных файлах серверов турниров для использования этой программы компиляции достаточно установить переменную `compile_dir` в следующее значение:

```
compile_dir = /var/ejudge/compile/var/compile
```

Переменная `run_dir`

Имя переменной: **`run_dir`**
Содержится в: `global`
Используются: serve, run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Значение по умолчанию: `run`
Может повторяться: *нет*

Описание. Данная переменная задаёт каталоги обмена между программами serve и run. Полный путь к этому каталогу определяется по следующим правилам:

- Если значение переменной `run_dir` не задано, используется значение `run`.
- Если значение переменной `run_dir` начинается с символа `'/'`, то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной `run_dir` не начинается с символа `'/'`, полный путь к каталогу скриптов образуется конкатенацией значения переменной `var_dir` и значения переменной `run_dir`.

Каталог обмена содержит в себе несколько подкаталогов, назначение которых описывается ниже.

- Подкаталог **`queue`** используется для передачи информации от программы serve к программе run. Программа serve помещает в этот каталог небольшие файлы, содержащие служебную информацию о программе, которую необходимо протестировать (так называемый «пакет задания тестирования»). Программа run периодически просматривает этот каталог и при появлении в нём новых пакетов задания тестирования выполняет их. Этот каталог может использоваться совместно несколькими программами serve, обслуживающими разные турниры, и несколькими программами run, например, работающими на разных компьютерах в сети. Этот каталог имеет специальную структуру, чтобы предотвратить синхронизационные ошибки при одновременном доступе нескольких программ.
- Подкаталог **`exe`** используется для передачи информации от программы serve к программе run. Программа serve помещает в этот каталог исполняемый файл программы, которую необходимо протестировать. Исполняемый файл программы имеет то же самое имя, что и пакет задания тестирования для этой программы. Программа run считывает исполняемый файл после файла пакета задания компиляции. Этот каталог может использоваться совместно несколькими программами serve, обслуживающими разные турниры, и несколькими программами run. Синхронизация доступа поддерживается в каталоге `queue`, поэтому никакой дополнительной синхронизации в каталоге `exe` не требуется.
- Подкаталог **`status`** используется для передачи информации от программы run к программе serve. В этот каталог программа run помещает небольшой файл с информацией о результате тестирования (так называемый «пакет результата тестирования»). Программа serve периодически просматривает данный каталог и при появлении в нём

новых файлов обновляет своё внутреннее состояние. Поскольку каталог может использоваться одновременно несколькими программами run и одной программой serve, он имеет специальную структуру для предотвращения синхронизационных ошибок.

- Подкаталог **report** используется для передачи информации от программы run к программе serve. В этот каталог программа run помещает судейский вариант протокола тестирования. Каталог может использоваться одновременно несколькими программами run и одной программой serve, но синхронизация доступа ведётся с помощью каталога status, поэтому никакой дополнительной синхронизации в каталоге report не требуется.

- Подкаталог **teamreport** используется для передачи информации от программы run к программе serve. В этот каталог программа run помещает пользовательский вариант протокола тестирования, если эта опция включена в конфигурационном файле турнира. Каталог может использоваться одновременно несколькими программами run и одной программой serve, но синхронизация доступа ведётся с помощью каталога status, поэтому никакой дополнительной синхронизации в каталоге teamreport не требуется.

- Символическая ссылка **<номер>**, где <номер> — четырёхзначный идентификатор турнира. Эта символическая ссылка создаётся в каталоге, задаваемом переменной run_dir, программы run, и указывает на каталог, задаваемый переменной compile_dir, программы serve. Если эти каталоги совпадают, символическая ссылка указывает на каталог, в котором она находится. С помощью этой символической ссылки программа run может обслуживать одновременно несколько турниров, записывая результаты тестирования в каталог обмена только соответствующего турнира.

Пример. С помощью задания каталога обмена можно добиться того, что несколько одновременно работающих серверов турнира будут использовать одну программу тестирования run. Для этого создаётся отдельный каталог для программы run, например /var/ejudge/run. Этот каталог указывается в качестве корневого в конфигурационном файле программы run с помощью строки

```
root_dir = /var/ejudge/run
```

Программа run запускается в каталоге /var/ejudge/run. Теперь во всех конфигурационных файлах серверов турниров для использования этой программы тестирования достаточно установить переменную run_dir в следующее значение:

```
run_dir = /var/ejudge/run/var/compile
```

Переменная serve_socket

Имя переменной: **serve_socket**
Содержится в: global
Используются: run
Тип содержимого: *путь к UNIX-сокету*
Может отсутствовать: *да*
Значение по умолчанию: serve
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к UNIX-сокету, который используется программой **serve** для взаимодействия с CGI-программами. Программа **serve** создаёт UNIX-сокет при запуске и удаляет при завершении, если только в командной строке не был задан ключ `-S`. Полный путь к сокету определяется по следующим правилам.

- Если значение переменной serve_socket не задано, используется значение serve.
- Если значение переменной serve_socket начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной serve_socket не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной var_dir и значения переменной serve_socket.

Внимание. Изменять значение данной конфигурационной переменной не рекомендуется.

Переменная run_log_file

Имя переменной: **run_log_file**
Содержится в: global
Используются: serve
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: run.log
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к файлу, в котором программа **serve** хранит базу посылок пользователей. Полный путь к файлу определяется по следующим правилам.

- Если значение переменной run_log_file не задано, используется значение run.log.
- Если значение переменной run_log_file начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к файлу, этот путь используется без изменений.
- Если значение переменной run_log_file не начинается с символа '/', полный путь к файлу образуется конкатенацией значения переменной var_dir и значения переменной run_log_file.

Внимание. Изменять значение данной конфигурационной переменной не рекомендуется.

Переменная clar_log_file

Имя переменной: **clar_log_file**
Содержится в: global
Используются: serve
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: clar.log
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к файлу, в котором программа serve хранит базу сообщений. Полный путь к файлу определяется по следующим правилам.

- Если значение переменной clar_log_file не задано, используется значение clar.log.
- Если значение переменной clar_log_file начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к файлу, этот путь используется без изменений.
- Если значение переменной clar_log_file не начинается с символа '/', полный путь к файлу образуется конкатенацией значения переменной var_dir и значения переменной clar_log_file.

Внимание. Изменять значение данной конфигурационной переменной не рекомендуется.

Переменная cr_serialization_key

Имя переменной: **cr_serialization_key**
Содержится в: global
Используются: compile, run
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: *нет*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить идентификатор SYSV IPC семафора, который будет использоваться для сериализации работы программ compile и run. Если эта конфигурационная переменная не установлена, сериализация запуска проводится не будет, то есть, если несколько программ compile и (или) несколько программ run работают на одной машине, возможен случай, когда все программы будут работать одновременно. Если сериализация включена, то не более одной программы будет выполняться в каждый момент времени.

Пример.

```
cr_serialization_key = 12345
```

Переменная `inactivity_timeout`

Имя переменной: **`inactivity_timeout`**
Содержится в: `global`
Используются: serve, run
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: `120`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная действует, если программа serve запущена с ключом `-S` в командной строке. Данный ключ передаёт программе serve файловый дескриптор UNIX-сокета, который будет использоваться как сокет команд программы serve. Этот аргумент командной строки используется, когда сервер турнира запускается по требованию программой super-serve. Конфигурационная переменная `inactivity_timeout` задаёт время в секундах, после которого программа serve завершает свою работу, если за указанный промежуток времени к ней не было ни одного обращения. Если значение конфигурационной переменной не установлено, предполагается значение 120 секунд (2 минуты). Если значение переменной установлено в 0, значение тайм-аута полагается неограниченным.

Для программы run данная конфигурационная переменная действует, если эта программа запущена с ключём `-S` в командной строке. Этот аргумент командной строки используется, когда сервер проверки решений запускается по требованию программой super-serve. Конфигурационная переменная `inactivity_timeout` задаёт время в секундах, после которого программа run завершает свою работу, если за указанный промежуток времени к ней не было ни одного обращения. Если значение конфигурационной переменной не установлено, предполагается значение 120 секунд (2 минуты). Если значение переменной установлено в 0, значение тайм-аута полагается неограниченным.

Пример.

```
inactivity_timeout = 300
```

Переменная `score_system`

Имя переменной: `score_system`
Содержится в: `global`
Используются: serve
Тип содержимого: *score_system*
Может отсутствовать: *да*
Значение по умолчанию: *ACM*
Может повторяться: *нет*

Описание. Данная переменная определяет тип турнира. Поддерживаются три типа турниров, описываемых ниже.

- Тип турнира *ACM*. Этот тип задаётся строкой

  ```
  score_system = acm
  ```

  в конфигурационном файле турнира. Данный тип турнира соответствует регламенту проведения чемпионата мира среди студенческих команд ACM. Он имеет следующие характеристики:

  1. Проверка решений ведётся непосредственно по ходу турнира. На каждую свою посылку участники немедленно получают ответ с результатами её тестирования.
  2. Участники ранжируются по числу решённых задач. Чем больше решено задач, тем выше место участника. Среди участников, решивших одинаковое количество задач, участники ранжируются по количеству штрафных очков. Чем меньше штрафных очков, тем выше место участника.
  3. Участник считается решившим задачу, если решение этой задачи, посланное участником, успешно прошло все тесты. Частичные решения не засчитываются.
  4. Штрафные очки для участника считаются как сумма штрафных очков по всем задачам, решённым участником. Нерешённые задачи не дают вклада в штрафные очки. Для решённой задачи штрафные очки по ней вычисляются как количество минут, прошедших от начала турнира до момента сдачи задачи, плюс по 20 минут за каждую предыдущую неуспешную попытку сдачи задачи. Попытки сдачи программы после успешной не учитываются.

- Тип турнира *KIROV*. Данный тип турниров придуман В. Матюхиным. Этот тип задаётся строкой

  ```
  score_system = kirov
  ```

  в конфигурационном файле турнира. Он имеет следующие характеристики:

  1. Проверка решений ведётся непосредственно по ходу турнира. На каждую свою посылку участники немедленно получают ответ с результатами её тестирования.
  2. Участники ранжируются по числу набранных баллов. Чем больше баллов набрал участник, тем выше его место.

3. Баллы, набранные участником, считаются как сумма баллов по каждой из задач турнира. Количество баллов, полученных участником за задачу, вычисляется как максимальное количество баллов, полученных участником за все попытки сдачи задачи.
4. Баллы, полученные участником за попытку вычисляются следующим образом:
   - Если решение участника прошло все тесты, оно получает полный балл за эту задачу (см. конфигурационную переменную `full_score`) за вычетом произведения штрафных баллов за попытку (см. конфигурационную переменную `run_penalty`) помноженных на количество предыдущих попыток сдачи решения. Если в результате получается число, меньшее нуля, количество баллов полагается равным нулю.
   - Если решение участника прошло часть тестов (либо не прошло ни одного теста), оно получает частичный балл за эту задачу в зависимости от конфигурационных переменных `test_score`, `test_score_list`, `test_sets` за вычетом произведения штрафных баллов за попытку (см. конфигурационную переменную `run_penalty`) помноженных на количество предыдущих попыток сдачи решения. Если в результате получается число, меньшее нуля, количество баллов полагается равным нулю.

- Тип турнира *OLYMPIAD*. Этот тип задаётся строкой

  ```
  score_system = olympiad
  ```

  в конфигурационном файле турнира. Данный тип турнира соответствует регламенту проведения российских и международных олимпиад по информатике. Он имеет следующие характеристики.

  1. Полная проверка решений участников ведётся после окончания основного времени турнира. Во время турнира присылаемые решения проверяются на нескольких первых тестах (см. конфигурационную переменную `tests_to_accept`). Если решение участника прошло все предварительные тесты, оно принимается для последующей проверки, а если решение не прошло хотя бы один тест, оно для проверки не принимается и в окончательной проверке решений по окончанию турнира не участвует.
  2. Участники ранжируются по числу набранных баллов. Чем больше баллов набрал участник, тем выше его место.
  3. Баллы, набранные участником, считаются как сумма баллов по каждой из задач турнира. Количество баллов, полученных участником за задачу, вычисляется как максимальное количество баллов, полученных участником за все попытки сдачи задачи.
  4. Баллы, полученные участником за попытку вычисляются следующим образом:
     - Если решение участника прошло все тесты, оно получает полный балл за эту задачу (см. конфигурационную переменную `full_score`) за вычетом произведения штрафных баллов за попытку (см. конфигурационную переменную `run_penalty`) помноженных на количество предыдущих попыток сдачи решения. Если в результате получается число, меньшее нуля, количество баллов полагается равным нулю.

– Если решение участника прошло часть тестов (либо не прошло ни одного теста), оно получает частичный балл за эту задачу в зависимости от конфигурационных переменных `test_score`, `test_score_list`, `test_sets` за вычетом произведения штрафных баллов за попытку (см. конфигурационную переменную `run_penalty`) помноженных на количество предыдущих попыток сдачи решения. Если в результате получается число, меньшее нуля, количество баллов полагается равным нулю.

Переменная `virtual`

Имя переменной: **`virtual`**
Содержится в: `global`
Используются: `serve`
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет устанавливать режим виртуального турнира. Виртуальный турнир поддерживается только для типа турнира *ACM*. Виртуальный турнир отличается тем, что отчёт времени для каждого участника ведётся независимо от других участников. Каждый участник получает возможность начать турнир в произвольный удобный для него момент времени. По истечении заданной продолжительности турнира (см. параметр `contest_time`) турнир для этого участника автоматически завершится.

Пример. Данный пример устанавливает режим виртуальности турнира.

```
virtual
```

Переменная `test_sfx`

Имя переменной: `test_sfx`
Содержится в: `global`
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает суффикс имён файлов, в которых хранятся входные данные тестов. Полный путь к очередному тесту определяется следующим образом:

```
path=${prob.test_dir}/${prob.short_name}/${test_num}${prob.test_sfx}
```

Здесь `prob` — это задача, решаемая тестируемой программой. `test_num` — это номер теста с тремя десятичными цифрами (включая ведущие незначащие нули). `${prob.test_dir}` — это значение конфигурационной переменной `test_dir` задачи. Если описание задачи не устанавливает эту переменную, используется глобальная переменная `test_dir`. `${prob.short_name}` — это короткое имя задачи (значение конфигурационной переменной `short_name`) описания задачи. `${prob.test_sfx}` — это значение конфигурационной переменной `test_sfx` описания задачи. Если в описании задачи эта переменная не установлена, используется глобальная переменная `test_sfx`.

Пример.

```
test_sfx = ".dat"
```

Переменная corr_sfx

Имя переменной: **corr_sfx**
Содержится в: global
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает суффикс имён файлов, в которых хранятся правильные ответы к тестам. Полный путь к очередному правильному ответу определяется следующим образом:

```
path=${prob.corr_dir}/${prob.short_name}/${test_num}${prob.corr_sfx}
```

Здесь prob — это задача, решаемая тестируемой программой. test_num — это номер теста с тремя десятичными цифрами (включая ведущие незначащие нули). ${prob.corr_dir} — это значение конфигурационной переменной corr_dir задачи. Если описание задачи не устанавливает эту переменную, используется глобальная переменная corr_dir. ${prob.short_name} — это короткое имя задачи (значение конфигурационной переменной short_name) описания задачи. ${prob.corr_sfx} — это значение конфигурационной переменной corr_sfx описания задачи. Если в описании задачи эта переменная не установлена, используется глобальная переменная corr_sfx.

Пример.

```
corr_sfx = ".res"
```

Переменная sound_player

Имя переменной: **sound_player**
Содержится в: global
Используются: run
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт полный путь к программе проигрывания звуковых файлов. Если переменная установлена, программа run проигрывает звуковые файлы, пути к которым установлены в конфигурационных переменных accept_sound, runtime_sound, timelimit_sound, wrong_sound, presentation_sound и internal_sound в зависимости от результата тестирования. Звуковое оповещение о результате тестирования поддерживается только в режиме турнира *ACM*.

Пример.

```
sound_player = /usr/bin/artsplay
```

Переменная `accept_sound`

Имя переменной: `accept_sound`
Содержится в: global
Используются: run
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если установлена данная конфигурационная переменная и установлена конфигурационная переменная sound_player, при успешном завершении тестирования, то есть если тестируемая программа успешно прошла все тесты, будет воспроизведён звуковой файл, путь к которому указан в данной конфигурационной переменной. Звуковое оповещение о результате тестирования поддерживается только в режиме турнира *ACM*.

Пример.

```
accept_sound = /usr/share/ejudge/sound/accept.wav
```

Переменная `runtime_sound`

Имя переменной: `runtime_sound`
Содержится в: global
Используются: run
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если установлена данная конфигурационная переменная и установлена конфигурационная переменная sound_player, при возникновении ошибки времени выполнения (runtime error) тестируемой программы будет воспроизведён звуковой файл, путь к которому указан в данной конфигурационной переменной. Звуковое оповещение о результате тестирования поддерживается только в режиме турнира *ACM*.

Пример.

```
runtime_sound = /usr/share/ejudge/sound/runtime.wav
```

Переменная `timelimit_sound`

Имя переменной: `timelimit_sound`
Содержится в: global
Используются: run
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если установлена данная конфигурационная переменная и установлена конфигурационная переменная sound_player, при превышении максимального времени, отведённого на работу тестируемой программы, (time limit exceeded) будет воспроизведён звуковой файл, путь к которому указан в данной конфигурационной переменной. Звуковое оповещение о результате тестирования поддерживается только в режиме турнира *ACM*.

Пример.

```
timelimit_sound = /usr/share/ejudge/sound/timelimit.wav
```

Переменная `wrong_sound`

Имя переменной: `wrong_sound`
Содержится в: global
Используются: run
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если установлена данная конфигурационная переменная и установлена конфигурационная переменная sound_player, если на некотором тесте тестируемая программа дала неправильный ответ (wrong answer), будет воспроизведён звуковой файл, путь к которому указан в данной конфигурационной переменной. Звуковое оповещение о результате тестирования поддерживается только в режиме турнира *ACM*.

Пример.

```
wrong_sound = /usr/share/ejudge/sound/wrong.wav
```

Переменная `presentation_sound`

Имя переменной: `presentation_sound`
Содержится в: global
Используются: run
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если установлена данная конфигурационная переменная и установлена конфигурационная переменная sound_player, если на некотором тесте тестируемая программа дала ответ, нарушающий формат результата, (presentation error), будет воспроизведён звуковой файл, путь к которому указан в данной конфигурационной переменной. Звуковое оповещение о результате тестирования поддерживается только в режиме турнира *ACM*.

Пример.

```
presentation_sound = /usr/share/ejudge/sound/presentation.wav
```

Переменная `internal_sound`

Имя переменной: `internal_sound`
Содержится в: global
Используются: run
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если установлена данная конфигурационная переменная и установлена конфигурационная переменная sound_player, и если при тестировании возникает внутренняя ошибка программы, проверяющей ответ тестируемой программы, будет воспроизведён звуковой файл, путь к которому указан в данной конфигурационной переменной. Звуковое оповещение о результате тестирования поддерживается только в режиме турнира *ACM*.

Пример.

```
internal_sound = /usr/share/ejudge/sound/internal.wav
```

Переменная autoupdate_standings

Имя переменной: **autoupdate_standings**
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: true
Может повторяться: *нет*

Описание. Данная конфигурационная переменная управляет автоматическим обновлением таблицы текущего положения участников турнира. Если данная конфигурационная переменная установлена в *true*, то таблицы текущего положения участников автоматически обновляются, когда становятся известными результаты тестирования очередной программы, а также в некоторых других случаях, за исключением времени «заморозки» (см. конфигурационные переменные board_fog_time, board_unfog_time). Кроме этого администратор турнира имеет возможность в любой момент обновить таблицу результатов. Если значение данной конфигурационной переменной установлено в *false* с помощью строки

```
autoupdate_standings = 0
```

то таблица текущего положения никогда не обновляется автоматически. Возможность обновлять таблицу текущих результатов с помощью программы-администратора турнира сохраняется.

Переменная standings_file_name

Имя переменной: **standings_file_name**
Содержится в: global
Используются: serve
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Значение по умолчанию: standings.html
Может повторяться: *нет*

Описание. Данная конфигурационная переменная определяет имя файла, в который выводится таблица текущего положения участников в турнире. Файл таблицы результатов таким образом создаётся с полным путём

```
${status_dir}/dir/${standings_file_name}
```

Где ${status_dir} — значение конфигурационной переменной status_dir. Чтобы сделать таблицу текущих результатов доступной для участников и зрителей, нужно создать символическую ссылку из дерева каталогов html-сервера на этот файл. При этом http-сервер должен поддерживать символические ссылки в том каталоге, в котором создана ссылка на таблицу текущих результатов. Например, для сервера Apache это достигается заданием опции FollowSymlinks в его конфигурационном файле или в файле .htaccess.

```
standings_file_name = "standings.shtml"
```

Переменная `stand_header_file`

Имя переменной: `stand_header_file`
Содержится в: global
Используются: serve
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная определена, то при генерации таблицы текущего положения участников в начало генерируемого файла будет помещаться текст, находящийся в файле-заголовке, полный путь к которому задаётся в этой конфигурационной переменной. Содержимое файла-заголовка считывается один раз при старте программы serve.

```
stand_header_file = "/home/httpd/ssi/stand_head.shtml"
```

Переменная `stand_footer_file`

Имя переменной: `stand_footer_file`
Содержится в: global
Используются: serve
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная определена, то при генерации таблицы текущего положения участников в конец генерируемого файла будет помещаться текст, находящийся в файле-подвале, полный путь к которому задаётся в этой конфигурационной переменной. Содержимое файла-подвала считывается один раз при старте программы serve.

```
stand_footer_file = "/home/httpd/ssi/stand_foot.shtml"
```

Переменная stand2_file_name

Имя переменной: **stand2_file_name**
Содержится в: global
Используются: serve
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Значение по умолчанию: *не установлена*
Может повторяться: *нет*

Описание. Если значение данной конфигурационной переменной определено, программа serve генерирует второй файл таблицы текущего положения участников турнира. Второй файл таблицы результатов создаётся со следующим полным путём:

```
${status_dir}/dir/${stand2_file_name}
```

Где ${status_dir} — значение конфигурационной переменной status_dir. Чтобы сделать таблицу текущих результатов доступной для участников и зрителей, нужно создать символическую ссылку из дерева каталогов html-сервера на этот файл. При этом http-сервер должен поддерживать символические ссылки в том каталоге, в котором создана ссылка на таблицу текущих результатов. Например, для сервера Apache это достигается заданием опции FollowSymlinks в его конфигурационном файле или в файле .htaccess. Необходимость двух отдельных файлов текущего положения команд обосновывается тем, что каждый из этих файлов может иметь независимое друг от друга дизайнерское оформление. Например, главный файл (см. переменную standings_file_name) может быть оформлен в контексте сайта олимпиады, а вспомогательный — чтобы демонстрироваться с помощью проектора.

```
stand2_file_name = "standings2.shtml"
```

Переменная stand2_header_file

Имя переменной: **stand2_header_file**
Содержится в: global
Используются: serve
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная и переменная stand2_file_name определены, то при генерации вспомогательной таблицы текущего положения участников в начало генерируемого файла будет помещаться текст, находящийся в файле-заголовке, полный путь к которому задаётся в этой конфигурационной переменной. Содержимое файла-заголовка считывается один раз при старте программы serve.

```
stand2_header_file = "/home/httpd/ssi/stand2_head.shtml"
```

Переменная `stand2_footer_file`

Имя переменной: `stand2_footer_file`
Содержится в: global
Используются: serve
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная и переменная stand2_file_name определены, то при генерации вспомогательной таблицы текущего положения участников в конец генерируемого файла будет помещаться текст, находящийся в файле-подвале, полный путь к которому задаётся в этой конфигурационной переменной. Содержимое файла-подвала считывается один раз при старте программы **serve**.

```
stand2_footer_file = "/home/httpd/ssi/stand2_foot.shtml"
```

Переменная `plog_file_name`

Имя переменной: `plog_file_name`
Содержится в: global
Используются: serve
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет задать имя файла, для генерируемого html-файла с журналом посылок. Журнал посылок содержит информацию о каждой посылке, которая включает в себя номер посылки, участника, задачу, язык и результат. Если значение переменной установлено, журнал посылок генерируется в файл со следующим путём:

```
${status_dir}/dir/${plog_file_name}
```

Где `${status_dir}` — значение конфигурационной переменной status_dir. Чтобы сделать журнал посылок доступным для участников и зрителей, нужно создать символическую ссылку из дерева каталогов html-сервера на этот файл. При этом http-сервер должен поддерживать символические ссылки в том каталоге, в котором создана ссылка на журнал посылок. Например, для сервера Apache это достигается заданием опции `FollowSymlinks` в его конфигурационном файле или в файле `.htaccess`.

```
plog_file_name = "log.shtml"
```

Переменная `plog_update_time`

Имя переменной: **`plog_update_time`**
Содержится в: `global`
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: `30`
Может повторяться: *нет*

Описание. Если установлена конфигурационная переменная `plog_file_name`, данная конфигурационная переменная позволяет задать период в секундах обновления журнала посылок турнира. В отличие от таблицы положения участников турнира, которая обновляется при поступлении новой информации, журнал посылок турнира обновляется всегда через один и тот же промежуток времени, задаваемый данной конфигурационной переменной.

Пример.

```
plog_update_time = 60
```

Переменная `plog_header_file`

Имя переменной: **`plog_header_file`**
Содержится в: `global`
Используются: serve
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная и переменная `plog_file_name` определены, то при генерации журнала посылок турнира в начало генерируемого файла будет помещаться текст, находящийся в файле-заголовке, полный путь к которому задаётся в этой конфигурационной переменной. Содержимое файла-заголовка считывается один раз при старте программы serve.

```
plog_header_file = "/home/httpd/ssi/plog_head.shtml"
```

Переменная `plog_footer_file`

Имя переменной: **`plog_footer_file`**
Содержится в: `global`
Используются: serve
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная и переменная `plog_file_name` определены, то при генерации журнала посылок турнира в конец генерируемого файла будет помещаться текст, находящийся в файле-подвале, полный путь к которому задаётся в этой конфигурационной переменной. Содержимое файла-подвала считывается один раз при старте программы serve.

```
plog_footer_file = "/home/httpd/ssi/plog_foot.shtml"
```

Переменная `team_info_url`

Имя переменной: `team_info_url`
Содержится в: global
Используются: serve
Тип содержимого: *шаблон URL*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт шаблон URL, по которому находится более подробная информация об участнике. Этот шаблон раскрывается в момент генерации таблицы положения участников в турнире и формирует гиперссылку от имени участника и его идентификатора к отдельной странице с подробной информацией о команде. Подробная информация о поддерживаемых форматных преобразованиях дана в разделе 2.11.2.

Пример.

```
team_info_url = "http://www.contest.ru/users/%Mi"
```

Переменная `prob_info_url`

Имя переменной: `prob_info_url`
Содержится в: global
Используются: serve
Тип содержимого: *шаблон URL*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт шаблон URL, по которому находится текст условия задачи. Этот шаблон раскрывается в момент генерации заголовка таблицы текущего положения участников в турнире и формирует гиперссылку от идентификатора задачи к отдельной странице с текстом этой задачи. Подробная информация о поддерживаемых форматных преобразованиях дана в разделе 2.11.2.

Пример.

```
prob_info_url = "http://www.contest.ru/problems/%lPs.shtml"
```

Переменная `team_enable_src_view`

Имя переменной: `team_enable_src_view`
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, участники турнира получают возможность просматривать текст программ, посланных ими на проверку и хранящихся в базе данных системы ejudge.

Пример.

```
team_enable_src_view
```

Переменная `team_enable_rep_view`

Имя переменной: `team_enable_rep_view`
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, участники турнира получают возможность просматривать сокращённую версию протокола тестирования для программ, посланных ими на проверку и хранящихся в базе данных системы ejudge. Сокращённая (пользовательская) версия протокола тестирования содержит только информацию о прохождении программой каждого теста, но не содержит самих тестов, результатов, выдаваемых программой, и правильных ответов.

Пример.

```
team_enable_rep_view
```

Переменная `report_error_code`

Имя переменной: `report_error_code`
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, и конфигурационная переменная team_enable_rep_view установлена в *true*, то сокращённая версия протокола тестирования будет содержать значение кода возврата, выработанное тестируемой программой на каждом тесте. В противном случае в сокращённом протоколе тестирования будет просто стоять отметка о том, был ли код завершения нормальным (0) или нет (любой ненулевой код). Информация о выработанном коде возврата может быть тривиальным образом использована для угадывания теста, поэтому по умолчанию эта информация отключена.

Пример.

```
report_error_code
```

Переменная `enable_continue`

Имя переменной: `enable_continue`
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, и продолжительность турнира неограничена (см. конфигурационную переменную contest_time), администратор турнира получает возможность продолжить турнир после того, так он был остановлен. Во всех других случаях остановленный или завершившийся турнир не может быть продолжен.

Пример.

```
enable_continue
```

Переменная `show_astr_time`

Имя переменной: `show_astr_time`
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, в таблице текущих посылок интерфейса администратора генерируемой программой master время посылки участником решения будет выведено как астрономическое время, а не как время от начала турнира. Данная конфигурационная переменная предназначена в первую очередь для использования в турнирах, продолжительность которых неограничена (см. конфигурационную переменную contest_time).

Пример.

```
show_astr_time
```

Переменная `team_download_time`

Имя переменной: `team_download_time`
Содержится в: global
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 30
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт, как часто участники турнира могут скачивать архив своих решений. Если значение переменной установлено в 0, загрузка архива своих решений участниками отключена.

Пример.

```
team_download_time = 600
```

Переменная `tests_to_accept`

Имя переменной: `tests_to_accept`
Содержится в: global
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 1
Может повторяться: *нет*

Описание. Данная конфигурационная переменная определяет, на ?сколько?? тестах должна быть проверена программа участника, чтобы быть принятой на проверку. Конфигурационная переменная действует только в режиме турнира *OLYMPIAD*. Во всех других режимах турнира её значение игнорируется. Если значение этой переменной установлено в 0, для принятия задачи на проверку прохождение ею тестов не требуется. Значение данной конфигурационной переменной может быть переопределено в описании задачи с помощью конфигурационной переменной tests_to_accept.

Пример.

```
tests_to_accept = 2
```

Переменная `disable_clars`

Имя переменной: `disable_clars`
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, возможность участников турнира и судей и администраторов писать сообщения друг другу отключена.

Пример.

```
disable_clars
```

Переменная `disable_team_clars`

Имя переменной: `disable_team_clars`
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, участники турнира не могут писать сообщения судьям, но судьи и администраторы турнира имеют возможность отправлять сообщения всем участникам или некоторому участнику.

Пример.

```
disable_team_clars
```

Переменная `ignore_compile_errors`

Имя переменной: `ignore_compile_errors`
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если значение данной конфигурационной переменной равно *true*, если программа, посланная участником, не может быть скомпилирована (например, из-за синтаксических ошибок), такая программа не засчитывается, то есть за неё не начисляется штраф.

Пример.

```
ignore_compile_errors
```

Переменная `ignore_duplicated_runs`

Имя переменной: `ignore_duplicated_runs`
Содержится в: global
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная имеет значение *true*, в последовательности из нескольких идущих подряд (для одного участника) посылок одной и той же программы для решения одной и той же задачи учитывается только первая посылка. Все остальные посылки игнорируются. Идентичность программ определяется с помощью сравнения их хэш-кода, полученного с помощью алгоритма SHA1.

Данная конфигурационная переменная может использоваться для удобства участников, работающих через очень медленный канал связи, так как нажатие кнопки “Reload” браузера в некоторых (редких) случаях может приводить к повторной посылке решения.

Пример.

```
ignore_duplicated_runs
```

Переменная `max_line_length`

Имя переменной: `max_line_length`
Содержится в: global
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: 4096
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт ограничение на максимальную длину строки в файле протокола тестирования программы. Если длина какой-либо строки в файле, который копируется в файл протокола тестирования, превышает это значение, такая строка заменяется на строку <string is too long>. Ограничение максимальной длины строки действует для файлов с входными данными тестов, файлов с правильными ответами к тестам, файлов с выводом тестируемой программы, файлов с выводом в поток ошибок тестируемой программы и программы, проверяющей правильность решения.

Пример.

```
max_line_length = 1024
```

Переменная `max_file_length`

Имя переменной: **`max_file_length`**
Содержится в: global
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: `65535`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт максимальный размер файла, копируемого в протокол тестирования программы. Если длина какого-либо файла превышает этот предел, содержимое файла заменяется на строку «`<file is too long`». Ограничение максимальной длины файла действует для файлов с входными данными тестов, файлов с правильными ответами к тестам, файлов с выводом тестируемой программы, файлов с выводом в поток ошибок тестируемой программы и программы, проверяющей правильность решения.

Пример.

```
max_file_length = 131072
```

Переменная `auto_short_problem_name`

Имя переменной: **`auto_short_problem_name`**
Содержится в: global
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: `false`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить режим, в котором короткое имя задачи будет генерироваться по её идентификатору. Если данная конфигурационная переменная имеет значение *true*, для каждой задачи короткое имя задачи строится по идентификатору задачи как запись идентификатора в десятичной системе с пятью цифрами, включая незначащие нули. Так, для задачи с идентификатором задачи `53` будет автоматически сгенерировано короткое имя `00053`.

Пример.

```
auto_short_problem_name
```

Переменная `checker_real_time_limit`

Имя переменной: `checker_real_time_limit`
Содержится в: `global`
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: `30`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт ограничение астрономического времени на проверку результата работы тестируемой программы на очередном тесте. Другими словами, эта переменная ограничивает максимальное время работы проверяющей программы. В случае, если максимальное время работы проверяющей программы превышено, проверка программы участника завершается со статусом “Check failed” («Проверка не удалась»). Значение 0 означает отсутствие ограничения времени.

Пример.

```
checker_real_time_limit = 60
```

Переменная `compile_real_time_limit`

Имя переменной: `compile_real_time_limit`
Содержится в: `global`
Используются: compile
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: `30`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт ограничение астрономического времени на компиляцию программы участника. В случае, если максимальное время компиляции превышено, проверка программы участника завершается со статусом “Check failed” («Проверка не удалась»). Значение 0 означает отсутствие ограничения времени.

Пример.

```
compile_real_time_limit = 60
```

Переменная `show_deadline`

Имя переменной: **`show_deadline`**
Содержится в: `global`
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true* и описание некоторой задачи содержит крайнее время сдачи (параметр `deadline`), то крайнее время сдачи будет показано в меню выбора задачи, показываемое CGI-программой team.

Пример.

```
show_deadline
```

Переменная variant_map_file

Имя переменной: **variant_map_file**
Содержится в: global
Используются: serve
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к файлу, в котором находится отображение регистрационных имён участников турнира в номера вариантов для всех вариантных задач данного турнира (см. конфигурационную переменную variant_num). Если в турнире есть хотя бы одна вариантная задача, параметр variant_map_file должен быть установлен в имя корректного файла. Полный путь к файлу вариантов формируется по следующим правилам: если значение переменной variant_map_file начинается с символа '/', то есть является абсолютным путем, используется значение этой переменной. В противном случае полный путь к файлу вариантов получается конкатенацией значения конфигурационной переменной conf_dir и значения переменной variant_map_file.

Текущая версия системы ejudge поддерживает следующий формат файла отображения вариантов. Файл имеет следующую структуру.

```
<variant_map version="1">
VARIANT_MAP_LINE*
</variant_map>
```

Здесь первая и последние строки должны присутствовать точно в указанном виде, VARIANT_MAP_LINE — строка отображения вариантов для одного пользователя. Комментарии в файле начинаются с символа # и оканчиваются концом строки. Пустые строки в файле игнорируются. Файл вариантов может содержать произвольное количество строк отображения вариантов, однако участник, не упомянутый в файле отображения вариантов, теряет возможность сдавать вариантные задачи. Попытка сдачи им вариантной задачи завершится ошибкой недопустимой задачи. Каждый участник может быть упомянут в файле вариантов не более одного раза.

Строка отображения вариантов одного пользователя имеет следующий вид:

```
LOGIN VARIANT*
```

Здесь LOGIN — это регистрационное имя пользователя, VARIANT — номер варианта. Номера вариантов должны быть перечислены для всех вариантных задач турнира и в том же порядке, в котором заданы вариантные задачи. Количество номеров вариантов в строке отображения варинтов должно совпадать с количеством вариантных задач турнира. Вариант задачи — это целое число от 1 и до значения конфигурационной перемененной variant_num соответствующей задачи включительно.

Пример.

```
variant_map_file = "variant.map"
```

Если в турнире определены три вариантные задачи, каждая из которых имеет по 4 варианта, то следующий пример задает отображение вариантов для пользователей user1, user2, user3.

```
<variant_map version="1">
# Задачи: A, B, C
user1     1  3  2
user2     3  4  1
user3     4  2  4
</variant_map>
```

Переменная `disable_auto_testing`

Имя переменной: `disable_auto_testing`
Содержится в: `global`
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная установлена в *true*, автоматическое тестирование решений задач, посылаемых участниками, отключено. В этом случае решение участника получает статус “Accepted for testing” и не отправляется на компиляцию и тестирование. Администратор турнира должен явно запустить тестирование решения участника, выбрав пункт “Rejudge” для данного решения.

Данная конфигурационная переменная устанавливает значение, используемое в случае, если описание задачи не определяет конфигурационную переменную `disable_auto_testing` самостоятельно.

Пример.

```
disable_auto_testing
```

Переменная `enable_runlog_merge`

Имя переменной: `enable_runlog_merge`
Содержится в: `global`
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, для администратора турнира становятся доступными средства слияния журналов посылок. В этом случае появляется новый элемент управления, генерируемый программой master. Используя его, администратор задаёт имя файла, содержащего журнал посылок в формате XML. Этот журнал посылок далее пересылается серверу турнира, который сливает его с текущим журналом посылок. Импорт возможен, только если в текущий момент в турнире нет непроверенных посылок. При импорте корректно объединяются идентичные записи в текущем и импортируемом журналах. При этом если некоторая запись имеет флаг «локальной», её данные (результат тестирования, количество набранных баллов) имеют приоритет над данными импортируемой записи. Если импортируемая запись имеет флаг «авторитетной», то её данные имеют приоритет над локальными данными. Если некоторая запись не является ни локальной, ни авторитетной, локальные данные имеют приоритет.

В результате импорта идентификаторы посылок могут измениться. При этом идентификатор с большим номером всегда имеет время отправки не меньшее, чем идентификатор с меньшим номером. Если несколько посылок имеют одинаковое время отправки, они упорядочиваются в порядке возрастания идентификаторов пользователей. Проимпортированные записи помечаются символом * рядом с номером посылки.

При экспорте журнала посылок локальные записи помечаются как авторитетные, что позволяет по ходу турнира обмениваться текущими журналами посылок.

Переменная `max_cmd_length`

Имя переменной: **`max_cmd_length`**
Содержится в: `global`
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Значение по умолчанию: `256`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает максимальный размер аргументов командной строки для добавления в протокол тестирования. Если размер аргументов командной строки превышает это значение, они не добавляются в протокол тестирования.

Переменная info_dir

Имя переменной: **info_dir**
Содержится в: global
Используются: run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором хранится дополнительная информация к тестам для всех задач данного турнира. Сами файлы с дополнительной информацией находятся в подкаталогах этого каталога. Дополнительная информация для некоторой задачи с кратким именем A (задаваемом конфигурационной переменной short_name) находятся в подкаталоге A каталога дополнительной информации. Формат файлов с дополнительной информацией о тестах описывается в разделе 2.12. Полный путь к каталогу с дополнительной информацией определяется по следующим правилам:

- Если значение переменной info_dir не задано, используется значение info.
- Если значение переменной info_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу скриптов, этот путь используется без изменений.
- Если значение переменной info_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной conf_dir и значения переменной info_dir.

Пример. В следующем примере путь к каталогу тестов устанавливается в ${root_dir}/conf/../tests, где ${root_dir} — значение конфигурационной переменной root_dir. Этот путь эквивалентен пути ${root_dir}/tests. Таким образом, тесты к задачам и дополнительная информация о тестах находятся в одном каталоге.

```
info_dir = ../tests
```

Переменная tgz_dir

Имя переменной: **tgz_dir**
Содержится в: global
Используются: run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором хранятся архивы рабочих каталогов для тестирования программ в формате .tgz для всех задач данного турнира. Непосредственно архивы рабочих каталогов находятся в подкаталогах этого каталога. Архив рабочего каталога для некоторой задачи с кратким именем A (задаваемом конфигурационной переменной short_name) находятся в подкаталоге A каталога архивов. Полный путь к каталогу с дополнительной информацией определяется по следующим правилам:

- Если значение переменной tgz_dir не задано, используется значение info.
- Если значение переменной tgz_dir начинается с символа '/', то есть значение переменной задаёт полный абсолютный путь к каталогу архивов, этот путь используется без изменений.
- Если значение переменной tgz_dir не начинается с символа '/', полный путь к каталогу скриптов образуется конкатенацией значения переменной conf_dir и значения переменной tgz_dir.

Пример. В следующем примере путь к каталогу архивов устанавливается в ${root_dir}/conf/../tests, где ${root_dir} — значение конфигурационной переменной root_dir. Этот путь эквивалентен пути ${root_dir}/tests. Таким образом, тесты к задачам и архивы рабочего каталога тестирования находятся в одном каталоге.

```
tgz_dir = ../tests
```

Переменная `info_sfx`

Имя переменной: **`info_sfx`**
Содержится в: `global`
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: *.inf*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает суффикс файлов с дополнительной информацией о тестах. Эти файлы находятся в подкаталогах каталога, задаваемого конфигурационной переменной `info_dir`. Формат файла с дополнительной информацией о тесте описан в разделе 2.12.

Переменная `tgz_sfx`

Имя переменной: **`tgz_sfx`**
Содержится в: `global`
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: *.tgz*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает суффикс файлов с архивами рабочего каталога тестируемых программ. Эти файлы находятся в подкаталогах каталога, задаваемого конфигурационной переменной `tgz_dir`.

Переменная `prune_empty_users`

Имя переменной: `prune_empty_users`
Содержится в: `global`
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, участники турнира, которые не сделали ни одной попытки решения задачи, будут исключены из таблицы результатов. Если данная конфигурационная переменная установлена в *false* (значение по умолчанию), все участники турнира отображаются в таблице текущих результатов. Данная конфигурационная переменная может использоваться в турнирах с открытой регистрацией (в интернет-турнирах или виртуальных турнирах), чтобы исключить участников, которые зарегистрировались, но так и не приняли участие в турнире.

Переменная `stand_table_attr`

Имя переменной: `stand_table_attr`
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет определить дополнительные атрибуты элемента `<table>` таблицы текущих результатов. Значение данной переменной и элемент `table` не разделяются дополнительными пробелами, поэтому значение данной переменной должно начинаться с пробельного символа.

Пример.

```
stand_table_attr = " width=100%"
```

Такое значение данной конфигурационной переменной приведет к тому, что таблица текущих результатов будет открываться тегом

```
<table width=100%>
```

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная `stand_place_attr`

Имя переменной: `stand_place_attr`
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет определить дополнительные атрибуты столбца «место» (1-й столбец) таблицы результатов турнира. Значение данной переменной добавляется к элементу `<th>` заголовка столбца таблицы и к каждому элементу `<td>`, находящемуся в столбце «место». Значение данной переменной и элементы `td` или `th` не разделяются дополнительными пробелами, поэтому значение данной переменной должно начинаться с пробельного символа.

Пример.

```
stand_place_attr = " class=st_place"
```

Такое значение данной конфигурационной переменной приведёт к тому, что заголовок столбца «место» таблицы будет открываться тегом `<th class=st_place>`, а каждая ячейка первого столбца таблицы — тегом `<td class=st_place>`.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная `stand_team_attr`

Имя переменной: `stand_team_attr`
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет определить дополнительные атрибуты столбца «команда» (2-й столбец) таблицы результатов турнира. Значение данной переменной добавляется к элементу `<th>` заголовка столбца таблицы и к каждому элементу `<td>`, находящемуся в столбце «команда». Значение данной переменной и элементы `td` или `th` не разделяются дополнительными пробелами, поэтому значение данной переменной должно начинаться с пробельного символа.

Пример.

```
stand_team_attr = " class=st_team"
```

Такое значение данной конфигурационной переменной приведёт к тому, что заголовок столбца «команда» таблицы будет открываться тегом `<th class=st_team>`, а каждая ячейка второго столбца таблицы — тегом `<td class=st_team>`.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная `stand_prob_attr`

Имя переменной: `stand_prob_attr`
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет определить дополнительные атрибуты столбцов «задача» (каждой задаче турнира соответствует один столбец) таблицы результатов турнира. Значение данной переменной добавляется к элементу `<th>` заголовка столбцов таблицы и к каждому элементу `<td>`, находящемуся в столбцах «задача». Значение данной переменной и элементы `td` или `th` не разделяются дополнительными пробелами, поэтому значение данной переменной должно начинаться с пробельного символа.

Пример.

```
stand_prob_attr = " class=st_prob"
```

Такое значение данной конфигурационной переменной приведёт к тому, что заголовок столбцов «задача» таблицы будет открываться тегом `<th class=st_prob>`, а каждая ячейка столбцов задач таблицы — тегом `<td class=st_prob>`.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная stand_solved_attr

Имя переменной: **stand_solved_attr**
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет определить дополнительные атрибуты столбца «решённые задачи» таблицы результатов турнира. Значение данной переменной добавляется к элементу <th> заголовка столбцов таблицы и к каждому элементу <td>, находящемуся в столбцах «решённые задачи». Значение данной переменной и элементы td или th не разделяются дополнительными пробелами, поэтому значение данной переменной должно начинаться с пробельного символа.

Пример.

```
stand_solved_attr = " class=st_solved"
```

Такое значение данной конфигурационной переменной приведёт к тому, что заголовок столбцов «решённые задачи» таблицы будет открываться тегом <th class=st_solved>, а каждая ячейка столбцов задач таблицы — тегом <td class=st_solved>.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная stand_score_attr

Имя переменной: **stand_score_attr**
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет определить дополнительные атрибуты столбца «баллы» таблицы результатов турнира. Данный столбец присутствует только в турнирах, проводимых по системе *OLYMPIAD* или *KIROV*. Значение данной переменной добавляется к элементу <th> заголовка столбцов таблицы и к каждому элементу <td>, находящемуся в столбцах «баллы». Значение данной переменной и элементы td или th не разделяются дополнительными пробелами, поэтому значение данной переменной должно начинаться с пробельного символа.

Пример.

```
stand_score_attr = " class=st_score"
```

Такое значение данной конфигурационной переменной приведёт к тому, что заголовок столбцов «баллы» таблицы будет открываться тегом <th class=st_score>, а каждая ячейка столбцов задач таблицы — тегом <td class=st_score>.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная `stand_penalty_attr`

Имя переменной: `stand_penalty_attr`
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет определить дополнительные атрибуты столбца «штраф» таблицы результатов турнира. Данный столбец присутствует только в турнирах, проводимых по системе *ACM*. Значение данной переменной добавляется к элементу `<th>` заголовка столбцов таблицы и к каждому элементу `<td>`, находящемуся в столбцах «штраф». Значение данной переменной и элементы `td` или `th` не разделяются дополнительными пробелами, поэтому значение данной переменной должно начинаться с пробельного символа.

Пример.

```
stand_penalty_attr = " class=st_penalty"
```

Такое значение данной конфигурационной переменной приведёт к тому, что заголовок столбцов «штраф» таблицы будет открываться тегом `<th class=st_penalty>`, а каждая ячейка столбцов задач таблицы — тегом `<td class=st_penalty>`.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная `stand_time_attr`

Имя переменной: `stand_time_attr`
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет определить атрибуты, с которыми выводится время сдачи задачи в таблице результатов турнира по системе *ACM*. Значение данной переменной добавляется к элементу `<div>` элементов таблицы. Значение данной переменной и элементы `div` не разделяются дополнительными пробелами, поэтому значение данной переменной должно начинаться с пробельного символа.

Пример.

```
stand_time_attr = " class=st_time"
```

Такое значение данной конфигурационной переменной приведёт к тому, что время сдачи задачи будет открываться тегом `<div class=st_time>`.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная stand_extra_format

Имя переменной: **stand_extra_format**
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить содержимое столбца «дополнительная информация» таблицы результатов турнира. Вывод дополнительного столбца активируется, если значение данной конфигурационной переменной непусто. Значение данной переменной определяет форматную подстановку (см. раздел 2.11.2), выполняемую при генерации значений данного столбца.

Пример. Пример.

```
stand_extra_format = "%Mc, %Mo"
```

Переменная stand_extra_legend

Имя переменной: **stand_extra_legend**
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить название столбца дополнительной информации таблицы результатов турнира. Столбец дополнительной информации активируется, если значение конфигурационной переменной stand_extra_format непусто.

Пример.

```
stand_extra_legend = "Город, Страна"
```

Переменная stand_extra_attr

Имя переменной: **stand_extra_attr**
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет определить атрибуты, с которыми выводится столбец дополнительной информации в таблице результатов турнира. Столбец дополнительной информации активируется, если значение конфигурационной переменной stand_extra_format непусто. Значение данной переменной добавляется к

элементу <th> заголовка столбцов таблицы и к каждому элементу <td>, находящемуся в столбцах «дополнительная информация». Значение данной переменной и элементы td или th не разделяются дополнительными пробелами, поэтому значение данной переменной должно начинаться с пробельного символа.

Пример.

```
stand_extra_attr = " class=st_extra"
```

Такое значение данной конфигурационной переменной приведёт к тому, что заголовок столбцов «дополнительная информация» таблицы будет открываться тегом <th class=st_extra>, а каждая ячейка столбцов задач таблицы — тегом <td class=st_extra>.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная stand_self_row_attr

Имя переменной: stand_self_row_attr
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить дополнительные атрибуты строки таблицы результатов виртуального турнира, соответствующей текущему виртуальному участнику. Значение данной переменной добавляется к элементу <tr> этой строки.

Пример.

```
stand_self_row_attr = " bgcolor=#ffeeff"
```

Такое значение данной конфигурационной переменной приведёт к тому, что строка таблицы результатов будет открываться тегом <tr bgcolor=#ffeeff>.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная stand_v_row_attr

Имя переменной: stand_v_row_attr
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить дополнительные атрибуты строк таблицы результатов виртуального турнира, соответствующих виртуальным

участникам виртуального турнира. Значение данной переменной добавляется к элементу <tr> этой строки.

Пример.

```
stand_v_row_attr = " bgcolor=#eeffff"
```

Такое значение данной конфигурационной переменной приведёт к тому, что строки таблицы результатов будет открываться тегом <tr bgcolor=#eeffff>.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная stand_r_row_attr

Имя переменной: stand_r_row_attr
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить дополнительные атрибуты строк таблицы результатов виртуального турнира, соответствующих невиртуальным участникам виртуального турнира. Значение данной переменной добавляется к элементу <tr> этой строки.

Пример.

```
stand_r_row_attr = " bgcolor=#ffffee"
```

Такое значение данной конфигурационной переменной приведёт к тому, что строки таблицы результатов будет открываться тегом <tr bgcolor=#ffffee>.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (") в значении конфигурационной переменной.

Переменная stand_u_row_attr

Имя переменной: stand_u_row_attr
Содержится в: global
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить дополнительные атрибуты строк таблицы результатов виртуального турнира, соответствующих участникам виртуального турнира с неопределённым статусом. Значение данной переменной добавляется к элементу <tr> этой строки.

Пример.

```
stand_u_row_attr = " bgcolor=#ffffff"
```

Такое значение данной конфигурационной переменной приведёт к тому, что строки таблицы результатов будет открываться тегом `<tr bgcolor=#ffffff>`.

Обратите внимание, что в текущей версии системы ejudge, к сожалению, невозможно использовать знак «кавычки» (**"**) в значении конфигурационной переменной.

2.11.5 Параметры задачи

Каждая задача описывается в отдельной секции с именем `problem`. Система ejudge в настоящее время поддерживает до 100 задач. Чтобы увеличить максимальное число одновременно поддерживаемых задач необходимо изменить константу `MAX_PROBLEM` в файле `prepare.c`.

Все конфигурационные переменные, доступные в описании задачи, перечислены ниже.

Название	Стр.	Описание
abstract	151	Флаг абстрактной задачи.
checker_real_time_limit	167	Ограничение времени работы проверяющей программы.
corr_dir	154	Каталог с правильными ответами к тестам.
corr_sfx	156	Суффикс файлов с правильными ответами к тестам.
deadline	168	Крайний срок сдачи задачи.
disable_auto_testing	170	Отключение режима автоматической проверки решения.
full_score	163	Количество баллов за полное решение задачи.
id	149	Идентификатор задачи.
info_dir	173	Каталог файлов с дополнительной информацией о тестах.
info_sfx	175	Суффикс файлов с дополнительной информацией о тестах.
input_file	158	Имя входного файла для решений задачи.
long_name	150	«Длинное» имя задачи.
output_file	159	Имя выходного файла для решений задачи.
real_time_limit	162	Лимит астрономического времени на 1 тест для решения.
run_penalty	163	Штраф за попытку решения задачи.
short_name	150	«Короткое» имя задачи.
super	151	Имя наследуемой абстрактной задачи.
team_enable_rep_view	160	Флаг разрешения просмотра участниками протокола тестирования.
tester_id	149	Идентификатор задачи для передачи на тестирование.
test_dir	153	Каталог с тестами к задаче.
test_score	164	Количество баллов за один тест.
test_score_list	165	Спецификация баллов за тесты.
test_sets	166	Спецификация баллов за множество тестов.
test_sfx	155	Суффикс имён файлов с тестами.
tests_to_accept	162	Количество тестов для принятия задачи в режиме турнира *OLYMPIAD*.
tgz_dir	174	Каталог архивов рабочего каталога тестируемых программ.
tgz_sfx	175	Суффикс файлов архивов рабочего каталога тестируемых программ.

Продолжение на следующей странице

Название	Стр.	Описание
time_limit	161	Лимит виртуального процессорного времени на 1 тест для решения.
use_corr	160	Флаг использования правильных ответов к тестам.
use_info	171	Флаг использования файлов с дополнительной информацией о тестах.
use_stdin	157	Флаг использования стандартного потока ввода для входных данных.
use_stdout	157	Флаг использования стандартного потока вывода для ответа.
use_tgz	172	Флаг использования файлов архивов рабочего каталога тестируемой программы.
variant_num	169	Общее количество вариантов данной задачи.

Переменная `id`

Имя переменной: **`id`**
Содержится в: problem
Используются: serve, run
Тип содержимого: *integer*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает идентификатор задачи. Идентификатор задачи — это целое число в интервале от 1 до константы MAX_PROBLEM, которая определяется в исходном файле `prepare.c` системы **ejudge**. Идентификатор задачи присваивается только неабстрактным задачам. Абстрактные задачи не имеют идентификатора. Каждая неабстрактная задача должна иметь уникальный идентификатор. Таким образом, одновременно поддерживается до неабстрактных 100 задач. Для увеличения количества одновременно поддерживаемых задач система **ejudge** должна быть перекомпилирована с большим значением константы MAX_PROBLEM.

Если в описании задачи её идентификатор явно не указан, идентификатор задачи назначается автоматически. Автоматически назначаемый идентификатор на 1 больше идентификатора предыдущей неабстрактной задачи. Такая политика автоматического назначения идентификаторов задачи может привести к ошибке из-за повторного использования одного и того же идентификатора для разных задач.

Переменная `tester_id`

Имя переменной: **`tester_id`**
Содержится в: problem
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная переменная задаёт идентификатор задачи для сервера тестирования run, если этот идентификатор отличается от идентификатора задачи в сервере турнира **serve**. По умолчанию основной идентификатор задачи и идентификатор задачи для тестирования совпадают. Данная конфигурационная переменная может применяться, если сервер тестирования настроен на тестирование задач нескольких турниров одновременно. В этом случае идентификатор задачи в конфигурационном файле сервера тестирования и идентификатор задачи в конфигурационном файле сервера турнира могут не совпадать. Конфигурационная переменная `tester_id` позволяет в этом случае установить идентификатор задачи для тестирования в описании задачи.

Эта конфигурационная переменная не наследуется от абстрактного описания задачи, поэтому её использование в абстрактных задачах не имеет смысла.

```
tester_id = 20
```

Переменная short_name

Имя переменной: short_name
Содержится в: problem
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает короткое имя задачи. Для абстрактных задач короткое имя задачи должно быть установлено и должно быть уникально среди всех абстрактных задач. Для неабстрактных задач короткое имя должно быть установлено, кроме случая, когда значение глобальной конфигурационной переменной auto_short_problem_name установлено в *true*. В этом случае если короткое имя задачи не задано, оно будет сгенерировано автоматически из идентификатора задачи, используя форматное преобразование %05d функции snprintf. Короткое имя неабстрактной задачи не обязано быть уникальным. Оно может совпадать с другим коротким именем абстрактной или неабстрактной задачи.

Короткое имя неабстрактной задачи является, по сути, идентификатором задачи для участников турнира и зрителей. Короткое имя задачи отображается в таблице текущего положения участников, в журнале посылок, в меню сдачи программы на проверку CGI-программы team и т. д. Традиционно короткое имя задачи — это заглавная латинская буква A, B и т. д. для основных задач турнира и Z, Y для пробных задач турнира.

Короткое имя абстрактной задачи используется для идентификации абстрактной задачи при наследовании её свойств с помощью конфигурационной переменной super. Короткое имя не наследуется.

```
short_name = "A"
```

Переменная long_name

Имя переменной: long_name
Содержится в: problem
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает «длинное» имя задачи, то есть её содержательное название. Описание абстрактной задачи не может устанавливать длинное имя задачи, длинное имя задачи не наследуется. Если описание неабстрактной задачи не устанавливает длинное имя задачи, оно генерируется автоматически по шаблону "Problem %d" по идентификатору задачи.

```
long_name = "За пивОм"
```

Переменная `abstract`

Имя переменной: `abstract`
Содержится в: problem
Используются: serve, run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: `false`
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, текущее описание задачи является описанием абстрактной задачи. Абстрактная задача не может иметь идентификатора задачи (id) или длинного имени задачи (long_name), но должна иметь короткое имя (short_name), уникальное среди всех абстрактных задач. Абстрактная задаче не может наследовать свойства другой абстрактной задачи, то есть в абстрактных задачах запрещено использование конфигурационной переменной super. Значения конфигурационных переменных, установленные в некоторой абстрактной задаче, наследуются конкретной задачей, указавшей данную абстрактную задачу в значении своей конфигурационной переменной super.

Конкретная задача наследует следующие конфигурационные переменные из описания абстрактной задачи: checker_real_time_limit, corr_dir, corr_sfx, disable_auto_testing, full_score, info_dir, info_sfx, input_file, output_file, real_time_limit, run_penalty, team_enable_rep_view, test_dir, test_score, test_sfx, test_to_accept, tgz_dir, tgz_sfx, time_limit, use_corr, use_info, use_stdin, use_stdout, use_tgz, variant_num.

При этом при наследовании конфигурационных переменных corr_dir, info_dir, input_file, output_file, test_dir tgz_dir выполняется форматная подстановка (см. раздел 2.11.2).

Не наследуются следующие конфигурационные переменные абстрактной задачи: abstract, deadline, id, long_name, short_name, super, tester_id, test_score_list, test_sets.

Переменная `super`

Имя переменной: `super`
Содержится в: problem
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена, описание задачи наследует свойства от указанной абстрактной задачи. Значение данной конфигурационной переменной должно быть коротким именем абстрактной задачи.

Конфигурационные переменные, значения которых наследуются, перечислены в описании конфигурационной переменной abstract.

Пример.

```
super = "Generic_Files"
```

Переменная `test_dir`

Имя переменной: **`test_dir`**
Содержится в: problem
Используются: serve, run
Тип содержимого: *путь к каталогу* или *шаблон*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором находятся тесты для данной задачи. Полный путь к каталогу с тестами для каждой неабстрактной задачи определяется по следующим правилам:

1. Если конфигурационная переменная `test_dir` задачи неопределена, и эта задача наследует свойства некоторой абстрактной задачи, у которой конфигурационная переменная определена, то выполняется форматная подстановка с форматом, определяемым значением переменной `test_dir` этой абстрактной задачи, и результат помещается в переменную `test_dir` данной неабстрактной задачи.

2. Если после предыдущего шага конфигурационная переменная `test_dir` всё ещё неопределена, её значение устанавливается в короткое имя данной задачи (см. переменную `short_name`).

3. Если после предыдущего шага значение конфигурационной переменной не начинается с символа `'/'`, то есть не является абсолютным путём к каталогу, значение данной конфигурационной переменной добавляется к значению глобальной конфигурационной переменной `test_dir`, и результат помещается в конфигурационную переменную `test_dir` задачи. Таким образом, глобальная конфигурационная переменная `test_dir` содержит первые компоненты пути к каталогу тестов, а конфигурационная переменная `test_dir` описания задачи — последние компоненты пути к каталогу тестов.

Пример. Следующий пример отключает распределение тестов по подкаталогам каталога, определяемого глобальной переменной `test_dir`.

```
test_dir = "."
```

Следующий пример для описания абстрактной задачи задаёт использование подкаталога, имя которого получается преобразованием к строчным буквам короткого имени задачи (`short_name`), в каталоге, определяемом глобальной переменной `test_dir`, для тестов каждой задачи, наследующей свойства данной абстрактной задачи.

```
test_dir = "%lPs"
```

Переменная `corr_dir`

Имя переменной: **`corr_dir`**
Содержится в: problem
Используются: serve, run
Тип содержимого: *путь к каталогу* или *шаблон*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором правильные ответы к тестам для данной задачи. Полный путь к каталогу с правильными ответами к тестам для каждой неабстрактной задачи определяется по следующим правилам:

1. Если конфигурационная переменная `corr_dir` задачи неопределена, и эта задача наследует свойства некоторой абстрактной задачи *A*, у которой конфигурационная переменная определена, то выполняется форматная подстановка с форматом, определяемым значением переменной `corr_dir` абстрактной задачи *A*, и результат помещается в переменную `corr_dir` данной неабстрактной задачи.

2. Если после предыдущего шага конфигурационная переменная `corr_dir` всё ещё неопределена, её значение устанавливается в короткое имя данной задачи (см. переменную `short_name`).

3. Если после предыдущего шага значение конфигурационной переменной не начинается с символа `'/'`, то есть не является абсолютным путём к каталогу, значение данной конфигурационной переменной добавляется к значению глобальной конфигурационной переменной `corr_dir`, и результат помещается в конфигурационную переменную `corr_dir` задачи. Таким образом, глобальная конфигурационная переменная `corr_dir` содержит первые компоненты пути к каталогу тестов, а конфигурационная переменная `corr_dir` описания задачи — последние компоненты пути к каталогу тестов.

Пример. Следующий пример отключает распределение правильных ответов к тестам по подкаталогам каталога, определяемого глобальной переменной `corr_dir`.

```
corr_dir = "."
```

Следующий пример для описания абстрактной задачи задаёт использование подкаталога, имя которого получается преобразованием к строчным буквам короткого имени задачи (`short_name`), в каталоге, определяемом глобальной переменной `corr_dir`, для правильных ответов к тестам для каждой задачи, наследующей свойства данной абстрактной задачи.

```
corr_dir = "%lPs"
```

Переменная `test_sfx`

Имя переменной: **`test_sfx`**
Содержится в: `problem`
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт суффикс файлов с тестами к задаче. Полный путь к очередному тесту определяется следующим образом:

```
path=${test_dir}/${test_num}${test_sfx}
```

Здесь `${test_dir}` — значение конфигурационной переменной `test_dir`, `${test_num}` — номер теста, записанный с тремя цифрами, включая незначащие нули, `${test_sfx}` — значение данной конфигурационной переменной.

Значение переменной `test_sfx` определяется по следующим правилам:

1. Если описание задачи не устанавливает переменную `test_sfx`, но наследует свойства некоторой абстрактной задачи *A*, которая устанавливает переменную `test_sfx`, то используется значение, установленное в описании абстрактной задачи *A*.

2. Если после предыдущего шага значение переменной всё ещё неопределено, используется значение глобальной конфигурационной переменной `test_sfx`.

3. Если после предыдущего шага значение переменной всё ещё неопределено, значение полагается равным пустой строке .

Пример.

```
test_sfx = ".dat"
```

Переменная `corr_sfx`

Имя переменной: `corr_sfx`
Содержится в: problem
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт суффикс файлов с правильными ответами к тестам к задаче. Полный путь к очередному правильному ответу к тесту определяется следующим образом:

```
path=${corr_dir}/${test_num}${corr_sfx}
```

Здесь `${corr_dir}` — значение конфигурационной переменной corr_dir, `${test_num}` — номер теста, записанный с тремя цифрами, включая незначащие нули, `${corr_sfx}` — значение данной конфигурационной переменной.

Правильные ответы к тестам при тестировании решения участника используются, только если конфигурационная переменная use_corr установлена в значение *true*. В этом случае проверяющей программе третьим аргументом командной строки передаётся полный путь к файлу с правильным ответом к тесту. Если значение конфигурационной переменной `use_corr` установлено в *false*, то путь к каталогу с ответами к тестам никак не используется, и третий аргумент в тестирующую программу не передаётся.

Значение переменной `corr_sfx` определяется по следующим правилам:

1. Если описание задачи не устанавливает переменную `corr_sfx`, но наследует свойства некоторой абстрактной задачи *A*, которая устанавливает переменную `corr_sfx`, то используется значение, установленное в описании абстрактной задачи *A*.

2. Если после предыдущего шага значение переменной всё ещё неопределено, используется значение глобальной конфигурационной переменной corr_sfx.

3. Если после предыдущего шага значение переменной всё ещё неопределено, значение полагается равным пустой строке .

Пример.

```
corr_sfx = ".res"
```

Переменная `use_stdin`

Имя переменной: `use_stdin`
Содержится в: problem
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, решение данной задачи должно считывать входные данный со стандартного потока ввода. Если же конфигурационная переменная установлена в *false*, решение должно считывать входные данные из файла, задаваемого с помощью конфигурационной переменной input_file.

Если значение данной переменной в описании неабстрактной задачи неопределено, то наследуется значение переменной из описания соответствующей абстрактной задачи (если таковая есть). Если и после этого значение переменной неопределено, используется значение по умолчанию *false*.

Пример.

```
use_stdin
```

Переменная `use_stdout`

Имя переменной: `use_stdout`
Содержится в: problem
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, решение данной задачи должно выводить результат вычислений на стандартный поток вывода. Если же конфигурационная переменная установлена в *false*, решение должно выводить результат вычислений в файл, имя которого задаётся с помощью конфигурационной переменной output_file.

Если значение данной переменной в описании неабстрактной задачи неопределено, то наследуется значение переменной из описания соответствующей абстрактной задачи (если таковая есть). Если и после этого значение переменной неопределено, используется значение по умолчанию *false*.

Пример.

```
use_stdout
```

Переменная `input_file`

Имя переменной: **`input_file`**
Содержится в: problem
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: `input`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает имя файла, из которого программы-решения данной задачи должны считывать входные данные согласно условию задачи. Эта переменная используется только, если значение конфигурационной переменной `use_stdin` равно *false*. Значение конфигурационной переменной `input_file` устанавливается по следующим правилам:

1. Если значение данной переменной в описании неабстрактной задачи не установлено, и эта неабстрактная задача наследует свойства некоторой абстрактной задачи *A*, в описании которой данная переменная установлена, то используется значение переменной из описания абстрактной задачи *A*, при этом выполняется форматная подстановка.

2. Если после предыдущего шага значение переменной всё ещё не задано, используется значение по умолчанию `input`.

Пример. Данная строка позволяет установить имя входного файла для абстрактной или неабстрактной задачи в `input.txt`.

```
input_file = "input.txt"
```

Следующий пример, может устанавливать имя входного файла в описании абстрактной задачи. Для всех неабстрактных задач, наследующих свойства данной абстрактной задачи, имя входного файла будет зависеть от короткого имени неабстрактной задачи. Например, для задачи с коротким именем `A` имя входного файла будет установлено в `a.in`.

```
input_file = "%lPs.in"
```

Переменная `output_file`

Имя переменной: **`output_file`**
Содержится в: `problem`
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: `output`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает имя файла, в который программы-решения данной задачи должны записывать результат вычислений согласно условию задачи. Эта переменная используется только, если значение конфигурационной переменной `use_stdout` равно *false*. Значение конфигурационной переменной `output_file` устанавливается по следующим правилам:

1. Если значение данной переменной в описании неабстрактной задачи не установлено, и эта неабстрактная задача наследует свойства некоторой абстрактной задачи *A*, в описании которой данная переменная установлена, то используется значение переменной из описания абстрактной задачи *A*, при этом выполняется форматная подстановка.

2. Если после предыдущего шага значение переменной всё ещё не задано, используется значение по умолчанию `output`.

Пример. Данная строка позволяет установить имя входного файла для абстрактной или неабстрактной задачи в `output.txt`.

```
output_file = "output.txt"
```

Следующий пример, может устанавливать имя входного файла в описании абстрактной задачи. Для всех неабстрактных задач, наследующих свойства данной абстрактной задачи, имя входного файла будет зависеть от короткого имени неабстрактной задачи. Например, для задачи с коротким именем `A` имя входного файла будет установлено в `a.out`.

```
output_file = "%lPs.out"
```

Переменная `use_corr`

Имя переменной: `use_corr`
Содержится в: problem
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если значение данной конфигурационной переменной установлено в *true*, при проверке решения участника турнира используется файл с правильными ответами на тесты. Полный путь к этому файлу передаётся в качестве третьего параметра при вызове программы, сравнивающей ответ, полученный программой участника, и правильный ответ. Если значение данной переменной установлено в *false*, файлы с правильными ответами не используются. В этом случае программе проверки решения передаётся только два аргумента.

Если значение данной конфигурационной переменной в описании неабстрактной задачи не установлено, оно наследуется из абстрактной задачи (если таковая определена). Если и абстрактная задача не устанавливает значение переменной `use_corr`, то используется значение по умолчанию, равное *false*.

Переменная `team_enable_rep_view`

Имя переменной: `team_enable_rep_view`
Содержится в: problem
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, участники турнира получают возможность просматривать протокол тестирования своей программы. Для участников генерируется специальная версия протокола тестирования, которая не содержит тестовых данных, вывода проверяемой программы на тестовых данных, вывода проверяющей программы и т. д.

Значение данной переменной для неабстрактной задачи определяется по следующим правилам:

1. Если значение данной переменной неопределено, но абстрактная задача *A*, свойства которой наследуются данной задачей, устанавливает значение данной переменной, используется значение из описания абстрактной задачи *A*.
2. Если после предыдущего шага значение данной переменной всё ещё неопределено, используется значение глобальной конфигурационной переменной `team_enable_rep_view`, если оно определено.
3. Если после предыдущего шага значение данной переменной всё ещё неопределено, используется значение по умолчанию *false*.

Переменная `time_limit`

Имя переменной: **`time_limit`**
Содержится в: `problem`
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: `0`
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт процессорное виртуальное время в секундах, отведённое на работу проверяемой программы с одним тестом. Если по истечении этого времени программа не завершила работу, она убивается, и для данного теста проверяемой программы диагностируется ошибка “Time-limit exceeded”. Виртуальное процессорное время — это время, в течении которого процессор выполнял (в режиме пользователя или в режиме ядра) данную задачу. Время, затраченное в ожидании событий ввода/вывода в данном параметре не учитывается. Значение 0 данной конфигурационной переменной означает, что время выполнения тестируемой программы не ограничивается.

Данный параметр соответствует ограничению времени работы программы на одном тесте, которое указывается в условии задачи. Однако, если тестируемая программа не занимает процессор а находится в ожидании события ввода-вывода, например, выполняя чтение из стандартного потока ввода вместо чтения из файла или выполняя системный вызов `pause`, она не тратит виртуальное процессорное время. Поэтому реальное астрономическое время выполнения тестируемой программы может быть сколь угодно больше виртуального процессорного времени. Для ограничения реального астрономического времени используется конфигурационная переменная `real_time_limit`.

Если данная конфигурационная переменная неабстрактной задачи неопределена, её значение наследуется от абстрактной задачи, если абстрактная задача указана. Если и после этого значение переменной неопределено, устанавливается значение по умолчанию `0` (отсутствие ограничений на максимальное время выполнения).

Пример.

```
time_limit = 5
```

Переменная `real_time_limit`

Имя переменной: `real_time_limit`
Содержится в: problem
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: 0
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт ограничение на астрономическое время выполнения тестируемой программы на одном тесте. Ограничение астрономического времени выполнения программы должно быть больше ограничения виртуального процессорного времени (см. переменную time_limit). Оно должно учитывать время выполнения операций ввода-вывода и загруженность тестирующей машины.

Если данная конфигурационная переменная неабстрактной задачи неопределена, её значение наследуется от абстрактной задачи, если абстрактная задача указана. Если и после этого значение переменной неопределено, устанавливается значение по умолчанию 0 (отсутствие ограничений на максимальное время выполнения).

Пример.

```
real_time_limit = 30
```

Переменная `tests_to_accept`

Имя переменной: `tests_to_accept`
Содержится в: problem
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: 1
Может повторяться: *нет*

Описание. Данная конфигурационная переменная определяет, сколько тестов должны быть успешно пройдены программой для того, чтобы она была принята на проверке. Конфигурационная переменная действует только в режиме турнира *OLYMPIAD*. Во всех других режимах турнира её значение игнорируется. Если значение этой переменной установлено в 0, для принятия задачи на проверку прохождение ею тестов не требуется.

Если данная конфигурационная переменная неабстрактной задачи неопределена, её значение наследуется от абстрактной задачи, если абстрактная задача указана. Если после этого значение данной переменной неопределено, используется значение глобальной конфигурационной переменной tests_to_accept. Значение глобальной конфигурационной переменной устанавливается в 1 по умолчанию, поэтому если ни глобальная переменная, ни переменная описания задачи не определены, используется значение 1.

Пример.

```
tests_to_accept = 2
```

Переменная `full_score`

Имя переменной: `full_score`
Содержится в: problem
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: 25
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает количество баллов, начисляемых за полное решение задачи, то есть за решение, которое успешно прошло все тесты. Переменная используется только в режимах турнира *KIROV* и *OLYMPIAD*. Полное описание правил начисления баллов за решение задачи см. в описании конфигурационной переменной score_system.

Если данная конфигурационная переменная неабстрактной задачи неопределена, её значение наследуется от абстрактной задачи, если абстрактная задача указана. Если и после этого значение переменной неопределено, устанавливается значение по умолчанию 25.

Пример.

```
full_score = 50
```

Переменная `run_penalty`

Имя переменной: `run_penalty`
Содержится в: problem
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: 1
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает штраф за каждую попытку решения задачи. Например, количество баллов, полученное за пятую попытку сдачи некоторой задачи, будет на 4*run_penalty баллов меньше, чем если то же самое решение было послано с первого раза. Переменная используется только в режимах турнира *KIROV* и *OLYMPIAD*. Полное описание правил начисления баллов за решение задачи см. в описании конфигурационной переменной score_system.

Если данная конфигурационная переменная неабстрактной задачи неопределена, её значение наследуется от абстрактной задачи, если абстрактная задача указана. Если и после этого значение переменной неопределено, устанавливается значение по умолчанию 1.

Пример.

```
run_penalty = 2
```

Переменная `test_score`

Имя переменной: `test_score`
Содержится в: problem
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: 1
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает, сколько баллов получает решение участника за успешное прохождение одного теста. Баллы, полученные за успешно пройденные тесты, суммируются. Количество баллов для каждого теста может быть определено индивидуально с помощью конфигурационной переменной test_score_list. Количество баллов, назначаемых за пройденное множество тестов, может быть определено с помощью конфигурационной переменной test_sets. Суммарное количество баллов за все тесты задачи не может превосходить значения конфигурационной переменной full_score.

Данная конфигурационная переменная используется только в режиме турнира *KIROV* или *OLYMPIAD* (см. конфигурационную переменную score_system).

Если данная конфигурационная переменная неабстрактной задачи неопределена, её значение наследуется от абстрактной задачи, если абстрактная задача указана. Если и после этого значение переменной неопределено, устанавливается значение по умолчанию 1.

Пример.

```
test_score = 2
```

Переменная `test_score_list`

Имя переменной: `test_score_list`
Содержится в: problem
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *нет*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет устанавливать число баллов, назначаемое за успешное прохождение тестов, индивидуально для каждого теста. Данная конфигурационная переменная используется только в режиме турнира *KIROV* или *OLYMPIAD* (см. конфигурационную переменную score_system). Если число баллов, назначаемое за успешное прохождение некоторого теста, в конфигурационной переменной test_score_list не определяется, для этого теста используется значение конфигурационной переменной test_score. Количество баллов, назначаемых за пройденное множество тестов, может быть определено с помощью конфигурационной переменной test_sets.

Значение данной конфигурационной переменной test_score_list — это спецификация баллов за тесты, которая имеет следующий формат (грамматика записана в нотации EBNF).

```
test_score_list_spec = { test_score_spec } ;
test_score_spec = [ "[" test_number "]" ] test_score ;
test_number = NUMBER ;
test_score = NUMBER ;
```

Другими словами, спецификация баллов за тесты — это список спецификаций баллов за отдельные тесты. В спецификации баллов за тест в квадратных скобках может указываться номер теста. Если номер теста не указан, используется номер предыдущего теста, увеличенный на 1. Если номер теста не указан в первой спецификации баллов, спецификация относится к первому тесту.

Значение данной конфигурационной переменной **не наследуется**.

Пример. Первый пример устанавливает количество баллов за тесты с первого по пятый в 1, 2, 3, 4, 5 баллов соответственно. Количество баллов за оставшиеся тесты (если они имеются) задаётся с помощью конфигурационной переменной test_score.

```
test_score_list = "1 2 3 4 5"
```

Второй пример устанавливает следующее количество баллов: за первый тест — 3 балла, за 10 тест — 4 балла, за 11 тест — 5 баллов, за 15 тест — 6 баллов.

```
test_score_list = "3 [10] 4 5 [15] 6"
```

Переменная `test_sets`

Имя переменной: `test_sets`
Содержится в: problem
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *нет*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *да*

Описание. Данная конфигурационная переменная позволяет задавать количество баллов, которое получает решение задачи, давшее правильный ответ на заданном множестве тестов. Данная конфигурационная переменная используется только в режиме турнира *KIROV* или *OLYMPIAD* (см. конфигурационную переменную score_system).

Данная конфигурационная переменная призвана бороться с угадыванием ответов. Например, если по условию задачи требуется дать ответ в форме "YES" или "NO", решение участника может всегда давать ответ "YES" или всегда давать ответ "NO", и такое «решение» может получить частичный балл (иногда довольно большой). Тогда чтобы бороться с такими программами, множество ответов "YES" и множество ответов "NO" могут быть указаны, как множества тестов, оцениваемые в 0 баллов. Теперь чтобы получить ненулевой балл решение участника должно хотя бы один раз правильно ответить "YES" и хотя бы один раз правильно ответить "NO".

Каждое значение конфигурационной переменной test_sets задаёт одно множество тестов, но переменная test_sets может быть повторена в описании задачи несколько раз, задавая несколько множеств тестов. Значение данной конфигурационной переменной **не наследуется**.

Значением одной конфигурационной переменной test_sets является строка-спецификация множества тестов, имеющая следующий формат (грамматика записана в нотации EBNF).

```
test_set_spec = test_num_list "=" set_score ;
test_num_list = { test_score } ;
set_score = NUMBER ;
test_score = NUMBER ;
```

Другими словами, спецификация множества тестов состоит из перечисления номеров тестов, за которым после знака "=" следует число баллов, назначаемое за данное множество тестов. Такое правило срабатывает только если множество тестов, на которых решение участника дало правильный ответ, в точности совпадает с указанным множеством тестов.

Пример. В следующем примере за успешное прохождение только тестов 2, 4, 5 решение участника получает 1 балл, а за успешное прохождение тестов 1, 4, 6 — 2 балла.

```
test_sets = "2 4 5 = 1"
test_sets = "1 4 6 = 2"
```

Переменная checker_real_time_limit

Имя переменной: **checker_real_time_limit**
Содержится в: problem
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: 30
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт ограничение на астрономическое время проверки результата работы тестируемой программы. Другими словами, эта переменная ограничивает максимальное время работы проверяющей программы. В случае, если максимальное время работы проверяющей программы превышено, проверка программы участника завершается со статусом “Check failed” («Проверка не удалась»). Значение 0 означает отсутствие ограничения времени.

Если значение данной конфигурационной переменной в описании неабстрактной задачи неопределено, но данная неабстрактная задача наследует свойства некоторой абстрактной задачи *A*, которая определяет данную переменную, используется значение, определённое в описании абстрактной задачи *A*. Если после этого значение данной конфигурационной переменной всё ещё неопределено, используется значение глобальной конфигурационной переменной checker_real_time_limit. Значение последней по умолчанию равно 30, следовательно и значение конфигурационной переменной уровня задачи также равно 30.

Пример.

```
checker_real_time_limit = 60
```

Переменная `deadline`

Имя переменной: **`deadline`**
Содержится в: problem
Используются: run
Тип содержимого: *календарное время*
Может отсутствовать: *да*
Наследуется: *нет*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить крайнее время, после наступления которого прекращается приём указанной задачи. Данная конфигурационная переменная позволяет избирательно прекращать приём определённых задач в определённое время. Её использование имеет смысл только для турниров, продолжительность которых неограничена. Календарное время задаётся в одном из следующих форматов:
$Y/M/D\ \ H:I:S$,
$Y/M/D\ \ H:I$,
$Y/M/D\ \ H$,
$Y/M/D$,
где Y — год в диапазоне от 1970 до 2038, M — номер месяца в диапазоне от 1 до 12, D — номер дня в месяце в диапазоне от 1 до 31, H — час в диапазоне от 0 до 23, I — минуты в диапазоне от 0 до 59, S — секунды в диапазоне от 0 до 60. Значение отсутствующих компонент времени полагается равным 0.

Пример.

```
deadline = "2003/10/01 00:00:00"
```

Следующая запись полностью эквивалентна предыдущей:

```
deadline = "2003/10/01"
```

Переменная variant_num

Имя переменной: **variant_num**
Содержится в: problem
Используются: serve, run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает количество вариантов задачи. Если значение данной переменной не установлено или явно установлено в 0, текущая задача не является вариантной, то есть предусматривается только один набор тестов, правильных ответов и одна проверяющая программа. Невариантными являются, как правило, все задачи в турнирах.

В некоторых случаях (например, при проведении самостоятельных работ с автоматической проверкой решений) желательно, чтобы у разных учащихся были бы разные задачи. Для этого можно использовать вариантные задачи. Вариантная задача — это семейство задач, у которых совпадают все параметры, кроме каталога с тестами, каталога с правильными ответами и проверяющей программы. Вариант задачи — это целое число, начиная от 1. Переменная variant_num задает количество вариантов задачи, которые, таким образом, нумеруются 1, 2 и до variant_num.

Все параметры, необходимые для тестирования вариантной задачи, получаются из невариантных параметров следующим образом. Каталог с тестами для варианта *n* должен находиться по пути ${test_dir}-*n*, где ${test_dir} — значение конфигурационной переменной test_dir данной задачи. Другими словами, к пути к каталогу с тестами, который был бы у данной задачи, если бы она не была вариантной, приписывается номер варианта, отделенный знаком «минус». Каталог с правильными ответами к тестам для варианта *n* должен находиться по пути ${corr_dir}-*n*, где ${corr_dir} — значение конфигурационной переменной corr_dir данной задачи. Проверяющая программа должна располагаться по пути ${check_cmd}-*n*, где ${check_cmd} — значение конфигурационной переменной check_cmd данной задачи.

Если значение переменной variant_num в описании задачи не установлена, но данная задача наследует свойства абстрактной задачи, в которой значение переменной установлено, будет использовано значение переменной, установленное в описании абстрактной задачи.

Пример.

```
variant_num = 4
```

Переменная `disable_auto_testing`

Имя переменной: `disable_auto_testing`
Содержится в: problem
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Если значение данной конфигурационной переменной задачи установлено в *true*, автоматическое тестирование решений данной задачи отключено. Принимаемые программой serve решения добавляются в базу решений участников в состоянии “Accepted for Testing” и не передаются на компиляцию и тестирование. Администратор турнира может запустить компиляцию и тестирование решения, выбрав пункт меню “Rejudge”.

Если значение данной конфигурационной переменной не установлено, но описание данной задачи наследует свойства некоторой абстрактной задачи, в которой значение данной переменной установлено, используется значение из описания абстрактной задачи. Если после этого значение данной переменной всё равно неопределено, используется значение глобальной конфигурационной переменной disable_auto_testing.

Пример.

```
disable_auto_testing
```

Переменная `use_info`

Имя переменной: **`use_info`**
Содержится в: `problem`
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если значение данной конфигурационной переменной установлено в *true*, для запуска решений участников на проверку и для проверки решений используются файлы с дополнительной информацией о тесте. В противном случае никаких дополнительных файлов не используется. Файлы с дополнительной информацией о тестах содержат аргументы командной строки для запуска тестируемой программы, комментарии к тестам и т. д. Такие файлы находятся в каталоге, определяемом конфигурационной переменной `info_dir`, имеют стандартное базовое имя, получаемое форматным преобразованием `%03d` функций семейства `printf` из номера теста, и суффикс, определяемый конфигурационной переменной `info_sfx` (по умолчанию `.inf`). Формат файлов с дополнительной информацией описан в разделе 2.12.

В случае использования `.inf`-файлов проверяющая программа получает дополнительный аргумент командной строки, содержащий путь к `.inf`-файлу для данного теста.

Если значение данной конфигурационной переменной в описании неабстрактной задачи не установлено, оно наследуется из абстрактной задачи (если таковая определена).

Переменная `use_tgz`

Имя переменной: `use_tgz`
Содержится в: problem
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если значение данной конфигурационной переменной установлено в *true*, для запуска решений участников на проверку создаётся рабочий каталог, и тестируемая программа запускается в этом файле. Исходное состояние рабочего каталога получается разархивированием архивного файла. Такие архивные файлы находятся в каталоге, определяемом конфигурационной переменной `tgz_dir`, имеют стандартное базовое имя, получаемое форматным преобразованием `%03d` функций семейства `printf` из номера теста, и суффикс, определяемый конфигурационной переменной `tgz_sfx` (по умолчанию `.tgz`).

Для теста с номером i `.tgz`-архив должен содержать единственный каталог с именем полученным форматным преобразованием `%03d` из номера теста i. Кроме этого, в каталоге `tgz_dir` должно находиться разархивированное содержимое этого архива, то есть каталог с именем $\langle$`%03d`$, i\rangle$. При запуске тестируемой программы соответствующий `.tgz` архив разворачивается в каталоге тестирования (`check_dir`) и текущий каталог запускаемой программы устанавливается на этот подкаталог. Файл с кодом программы, файлы входных и выходных данных таким образом оказываются на один уровень выше текущего каталога.

В случае использования `.tgz`-файлов проверяющая программа получает дополнительный аргумент командной строки, содержащий путь к развернутому содержимому соответствующего `.tgz` файла, то есть к каталогу с именем $\langle$`%03d`$, i\rangle$ в каталоге, определяемом конфигурационной переменной `tgz_dir`.

Если значение данной конфигурационной переменной в описании неабстрактной задачи не установлено, оно наследуется из абстрактной задачи (если таковая определена).

Переменная info_dir

Имя переменной: **info_dir**
Содержится в: problem
Используются: run
Тип содержимого: *путь к каталогу* или *шаблон*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором находится дополнительная информация о тестах к данной задаче. Файлы с дополнительной информацией о тестах далее кратко называются `.inf`-файлы. Полный путь к каталогу с `.inf`-файлами для каждой неабстрактной задачи определяется по следующим правилам:

1. Если конфигурационная переменная `info_dir` задачи неопределена, и эта задача наследует свойства некоторой абстрактной задачи *A*, у которой эта конфигурационная переменная определена, то выполняется форматная подстановка с форматом, определяемым значением переменной `info_dir` абстрактной задачи *A*, и результат помещается в переменную `info_dir` данной неабстрактной задачи.

2. Если после предыдущего шага конфигурационная переменная `info_dir` всё ещё неопределена, её значение устанавливается в короткое имя данной задачи (см. переменную `short_name`).

3. Если после предыдущего шага значение конфигурационной переменной не начинается с символа `'/'`, то есть не является абсолютным путём к каталогу, значение данной конфигурационной переменной добавляется к значению глобальной конфигурационной переменной `info_dir`, и результат помещается в конфигурационную переменную `info_dir` задачи. Таким образом, глобальная конфигурационная переменная `info_dir` содержит первые компоненты пути к каталогу `.inf`, а конфигурационная переменная `info_dir` описания задачи — последние компоненты пути к каталогу `.inf`-файлов.

Пример. Следующий пример отключает распределение `.inf`-файлов по подкаталогам каталога, определяемого глобальной переменной `info_dir`.

```
info_dir = "."
```

Следующий пример для описания абстрактной задачи задаёт использование подкаталога, имя которого получается преобразованием к строчным буквам короткого имени задачи (`short_name`), в каталоге, определяемом глобальной переменной `info_dir`, для `.inf`-файлов для каждой задачи, наследующей свойства данной абстрактной задачи.

```
info_dir = "%lPs"
```

Переменная tgz_dir

Имя переменной: **tgz_dir**
Содержится в: problem
Используются: run
Тип содержимого: *путь к каталогу* или *шаблон*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к каталогу, в котором находится архив рабочего каталога для тестирования решения участника. Архивы рабочих каталогов далее кратко называются `.tgz`-файлы. Полный путь к каталогу с `.tgz`-файлами для каждой неабстрактной задачи определяется по следующим правилам:

1. Если конфигурационная переменная `tgz_dir` задачи неопределена, и эта задача наследует свойства некоторой абстрактной задачи *A*, у которой эта конфигурационная переменная определена, то выполняется форматная подстановка с форматом, определяемым значением переменной `tgz_dir` абстрактной задачи *A*, и результат помещается в переменную `tgz_dir` данной неабстрактной задачи.

2. Если после предыдущего шага конфигурационная переменная `tgz_dir` всё ещё неопределена, её значение устанавливается в короткое имя данной задачи (см. переменную `short_name`).

3. Если после предыдущего шага значение конфигурационной переменной не начинается с символа `'/'`, то есть не является абсолютным путём к каталогу, значение данной конфигурационной переменной добавляется к значению глобальной конфигурационной переменной `tgz_dir`, и результат помещается в конфигурационную переменную `tgz_dir` задачи. Таким образом, глобальная конфигурационная переменная `tgz_dir` содержит первые компоненты пути к каталогу `.tgz`-файлов, а конфигурационная переменная `tgz_dir` описания задачи — последние компоненты пути к каталогу `.tgz`-файлов.

Пример. Следующий пример отключает распределение `.tgz`-файлов по подкаталогам каталога, определяемого глобальной переменной `tgz_dir`.

```
tgz_dir = "."
```

Следующий пример для описания абстрактной задачи задаёт использование подкаталога, имя которого получается преобразованием к строчным буквам короткого имени задачи (`short_name`), в каталоге, определяемом глобальной переменной `tgz_dir`, для `.tgz`-файлов для каждой задачи, наследующей свойства данной абстрактной задачи.

```
tgz_dir = "%lPs"
```

Переменная `info_sfx`

Имя переменной: **`info_sfx`**
Содержится в: problem
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает суффикс файлов с дополнительной информацией о тестах. Если неабстрактная задача не устанавливает непосредственно данную переменную, но наследует абстрактную задачу, устанавливающую эту конфигурационную переменную, используется значение из описания абстрактной задачи. Если после этого значение конфигурационной переменной `info_sfx` всё ещё неопределено, используется значение глобальной конфигурационной переменной `info_sfx`, которая по умолчанию равна `.inf`. Таким образом можно считать, что файлы с дополнительной информацией о тесте имеют по умолчанию суффикс `.inf`.

Переменная `tgz_sfx`

Имя переменной: **`tgz_sfx`**
Содержится в: problem
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает суффикс файлов архивов рабочего каталога для тестирования программ. Если неабстрактная задача не устанавливает непосредственно данную переменную, но наследует абстрактную задачу, устанавливающую эту конфигурационную переменную, используется значение из описания абстрактной задачи. Если после этого значение конфигурационной переменной `tgz_sfx` всё ещё неопределено, используется значение глобальной конфигурационной переменной `tgz_sfx`, которая по умолчанию равна `.tgz`. Таким образом можно считать, что файлы с архивами рабочего каталога имеют по умолчанию суффикс `.tgz`.

2.11.6 Параметры языка

Каждый поддерживаемый язык программирования описывается в секции с именем `language`. Система ejudge в нестоящее время поддерживает до 31 языка программирования. Чтобы увеличить количество одновременно поддерживаемых языков программирования необходимо изменить константу `MAX_LANGUAGE` в файле `prepare.c` в исходных текстах системы и перекомпилировать систему.

Название секции `language` можно считать не очень удачным, так как каждая секция описывает не язык программирования, а компилятор языка программирования. Например, конфигурационный файл может содержать 4 секции, соответствующие языку Паскаль: для GNU Pascal, Free Pascal, Delphi, Borland Pascal.

Для секций описания языка наследование свойств не поддерживается.

Все конфигурационные переменные, допустимые в секции описания языка программирования, приведены ниже.

Название	Стр.	Описание
arch	180	Архитектура компилятора.
cmd	181	Команда запуска компилятора.
compile_dir	182	Каталог обмена программ serve и compile.
compile_id	177	Идентификатор языка для передачи программе compile.
compile_real_time_limit	182	Ограничение времени на работу компилятора.
disabled	178	Флаг запрета использования данного компилятора.
exe_sfx	181	Суффикс исполняемых файлов, генерируемых компилятором.
id	177	Идентификатор языка.
key	179	Ключ (дополнительный параметр выбора).
long_name	179	Длинное (полное) имя компилятора.
short_name	178	Короткое имя компилятора.
src_sfx	180	Суффикс исходных файлов для компилятора.

Переменная `id`

Имя переменной: **`id`**
Содержится в: `language`
Используются: serve, compile
Тип содержимого: *integer*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает идентификатор языка программирования. Идентификатор является целым числом от 1 и до максимального количества поддерживаемых языков, задаваемого константой `MAX_LANGUAGE` в исходном файле `prepare.c` (значение в стандартной поставке — 31). Никакие два языка программирования не могут иметь одинаковые идентификаторы. Если идентификатор явно не задан, он назначается автоматически. Для этого берётся идентификатор предыдущего языка и увеличивается на 1. Если при этом получится уже использованный идентификатор, диагностируется фатальная ошибка. Если идентификатор первого языка не задан, он устанавливается равным 1.

Пример.

```
id = 10
```

Переменная `compile_id`

Имя переменной: **`compile_id`**
Содержится в: `language`
Используются: serve
Тип содержимого: *integer*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить дополнительный идентификатор языка для передачи решения участника на компиляцию. Переменная предназначена для случаев, когда одна и та же программа compile обслуживает сразу несколько турниров (см. также конфигурационную переменную `compile_dir`). В этом случае идентификатор языка программирования для сервера компиляции compile может отличаться от идентификатора языка сервера турнира serve. Данный конфигурационный параметр используется программой serve и позволяет задать идентификатор языка, соответствующий идентификатору языка у программы compile. Если данная переменная не указана, она полагается равной значению переменной `id`, то есть в случае, если serve и compile используют один конфигурационный файл, установка переменной `compile_id` не требуется.

Пример.

```
id = 10
```

Переменная `disabled`

Имя переменной: **`disabled`**
Содержится в: `language`
Используются: serve
Тип содержимого: *boolean*
Может отсутствовать: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, то участники не могут использовать данный язык программирования для посылки решений. Данный параметр может оказаться полезным, когда по ходу турнира необходимо запретить какой-либо язык программирования. Если просто удалить соответствующую запись, то, во-первых, могут измениться идентификаторы оставшихся языков, что приведёт к проблемам с базой решений участников турнира, и, во-вторых, возникнут проблемы с решениями участников, уже сданных на удалённом языке (вторичный ключ «язык программирования» в базе решений участников турнира будет указывать «в никуда»). Поэтому в таких случаях следует устанавливать конфигурационную переменную `disabled`.

Пример.

```
disabled
```

Переменная `short_name`

Имя переменной: **`short_name`**
Содержится в: `language`
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает «короткое имя» языка программирования. Короткое имя используется в таблице решений участников в CGI-программе master, в таблице решений в CGI-программе team, а также для выбора языка программирования для отсылки решения. Если эта конфигурационная переменная не установлена, она автоматически устанавливается в значение `lang<id>`, где `<id>` — это идентификатор задачи, преобразованный в строку с помощью спецификации формата `%d`.

Пример.

```
short_name = "gcc"
```

Переменная `long_name`

Имя переменной: `long_name`
Содержится в: language
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает «длинное имя» языка программирования. Длинное имя может содержать дополнительную информацию о языке программирования, например, версию компилятора. CGI-программа team использует значение этой переменной в меню выбора языка для отсылки решения. Если эта конфигурационная переменная не установлена, она автоматически устанавливается в значение "Language <id>", где <id> — это идентификатор задачи, преобразованный в строку с помощью спецификации формата %d.

Пример.

```
long_name = "GNU C 3.3.1"
```

Переменная `key`

Имя переменной: `key`
Содержится в: language
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает «ключ» для выбора языков программирования, обслуживаемых программой compile. Программа compile поддерживает опцию командной строки -k с помощью которой можно указать ключ языков программирования. Тогда из всех языков программирования, описанных в конфигурационном файле, будут обслуживаться только те, ключ которых совпадает с ключём, заданным в командной строке. В этом случае значение по умолчанию (пустая строка) переменной key является значимым, и такой язык программирования будет обслуживаться, только если в командной строке в качестве параметра опции -k указана пустая строка. Если опция -k не задана, отбор языков по значению переменной key не производится, и значение этой переменной может быть произвольным (в частности, переменная может быть не установлена).

Пример.

```
key = "langs_1"
```

Переменная `arch`

Имя переменной: **`arch`**
Содержится в: language
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт «архитектуру», для которой предназначен данный язык программирования. Архитектура является одним из двух параметров, по которому выбирается тестировщик для данной программы. Например, значение архитектуры `dos` может соответствовать компиляторам для DOS (например, Borland C 3.1), а неустановленное значение архитектуры (пустая строка) может соответствовать компиляторам для Linux, которые генерируют статический исполняемый модуль (например, GCC). Процедура запуска скомпилированной программы для этих архитектур существенно отличается, поэтому для каждой из них должен использоваться отдельный тестировщик. Исполняемый файл, полученный в результате компиляции исходного файла, будет направлен на тестирование к тестировщику с соответствующей архитектурой.

Пример.

```
arch = "dos"
```

Переменная `src_sfx`

Имя переменной: **`src_sfx`**
Содержится в: language
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает суффикс, который должны иметь файлы, подаваемые на компиляцию данному компилятору языка программирования. Например, компилятор GCC требует, чтобы файл программы на языке Си имел суффикс `.c`. а файл программы на языке Си++ имел суффикс `.cpp`, `.cc` или `.C`.

Пример.

```
src_sfx = ".c"
```

Переменная exe_sfx

Имя переменной: **exe_sfx**
Содержится в: language
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: "" (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает суффикс исполняемых файлов, получаемых в результате компиляции данным компилятором. Как правило, суффикс исполняемых файлов зависит от архитектуры. Например, исполняемые файлы для Linux обычно не имеют никакого суффикса, а исполняемые файлы для DOS имеют суффикс .exe.

Пример.

```
exe_sfx = ".exe"
```

Переменная cmd

Имя переменной: **cmd**
Содержится в: language
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *нет*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает программу, которая запускается для компиляции решения участника турнира. Как правило, эта программа не является непосредственно компилятором, а представляет собой скрипт, запускающий компилятор с необходимыми параметрами.

Программа компиляции, задаваемая этой переменной, запускается с двумя параметрами: первый параметр — имя исходного файла программы, второй параметр — имя файла, в который должен быть помещён результат компиляции. Программа запускается с текущим каталогом, установленным на рабочий каталог, в котором находится исходный файл, и должен находиться файл-результат. Стандартный поток вывода и стандартный поток ошибок перенаправляются в файл, который в случае неудачи компиляции будет использоваться как протокол запуска решения пользователя. Программа компиляции должна завершаться с кодом возврата 0, если компиляция завершилась успешно, и с любым другим кодом возврата в случае ошибки при компиляции.

Если значение конфигурационной переменной cmd начинается с символа '/', то есть задаёт абсолютный путь к программе, то используется этот путь. В противном случае полный путь к программе компиляции получается добавлением значения данной конфигурационной переменной cmd к значению глобальной конфигурационной переменной script_dir. Файл программы компиляции должен допускать выполнение, то есть иметь установленным бит x прав доступа.

Пример.

```
cmd = "gcc"
```

Переменная `compile_dir`

Имя переменной: `compile_dir`
Содержится в: language
Используются: serve
Тип содержимого: *string*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить каталог обмена программ serve и compile для данного языка программирования. Если переменная не установлена, используется значение глобальной переменной compile_dir. Данная переменная может использоваться, когда необходимо явно назначить сервер компиляции для некоторого языка программирования. При обычном использовании, когда один сервер компиляции поддерживает компиляцию всех определённых в турнире языков программирования, данная конфигурационная переменная не должна использоваться.

Переменная `compile_real_time_limit`

Имя переменной: `compile_real_time_limit`
Содержится в: language
Используются: compile
Тип содержимого: *integer*
Может отсутствовать: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт ограничение астрономического времени на компиляцию программы участника. В случае, если максимальное время компиляции превышено, проверка программы участника завершается со статусом “Check failed” («Проверка не удалась»). Значение 0 означает отсутствие ограничения времени. Если данная конфигурационная переменная не установлена, используется значение глобальной конфигурационной переменной compile_real_time_limit.

Пример.

```
compile_real_time_limit = 60
```

2.11.7 Параметры тестировщика

Разнообразные архитектурно-зависимые параметры тестирования решений задач описываются в секции тестировщика `tester`. Для каждой пары задача-архитектура должен быть описан тестировщик. Все параметры, относящиеся непосредственно к задаче, описываются в секции `problem`. Ключём для выбора тестировщика являются либо идентификатор задачи, задаваемый с помощью конфигурационной переменной `id`, либо короткое имя задачи, задаваемое с помощью конфигурационной переменной `short_name`. Архитектура определяется конфигурационной переменной `arch` секции описания языка программирования `language`. При этом несколько языков программирования могут иметь одну и ту же архитектуру. Поскольку секция `language` описывает, скорее, не язык программирования, а компилятор языка программирования, то и параметр `arch` описывает архитектуру, для которой генерирует код данный компилятор.

В секции тестировщика поддерживается наследование свойств неабстрактного тестировщика от абстрактного тестировщика аналогично тому, как наследуются свойства в секции задачи. При наследовании свойств в некоторых случаях выполняются форматные подстановки. Допускаются тестировщики по умолчанию, которые предназначены для проверки всех задач, не проверяемых явно заданными тестировщиками, для некоторой архитектуры. В тестировщиках по умолчанию могут использоваться форматные подстановки.

Текущая версия системы ejudge поддерживает до 100 небстрактных тестировщиков одновременно. Каждый тестировщик по умолчанию считается за один тестировщик независимо от количества задач, для которых он предназначен. Ограничение на максимальное количество тестировщиков задаётся константой `MAX_TESTER` в исходном файле `prepare.c` системы ejudge. Чтобы изменить это ограничение нужно изменить значение константы и перекомпилировать систему ejudge.

Все конфигурационные переменные тестировщика перечислены ниже.

Название	Стр.	Описание
`abstract`	187	Флаг абстрактного тестировщика.
`any`	186	Флаг тестировщика по умолчанию.
`arch`	189	Архитектура тестировщика.
`check_cmd`	202	Путь к программе проверки ответа.
`check_dir`	199	Рабочий каталог тестирования.
`clear_env`	196	Флаг очистки переменных окружения перед запуском тестируемой программы.
`errorcode_file`	200	Файл, содержащий код завершения тестируемой программы.
`error_file`	200	Файл, содержащий вывод на стандартный поток ошибок тестируемой программы.
`id`	184	Идентификатор тестировщика.
`is_dos`	191	Флаг DOS-формата текста для тестируемых программ.
`key`	192	Дополнительный параметр выбора — «ключ».
`kill_signal`	193	Сигнал принудительного завершения тестируемой программы.
`max_data_size`	195	Максимальный размер данных тестируемой программы.

Продолжение на следующей странице

Название	Стр.	Описание
max_stack_size	194	Максимальный размер стека тестируемой программы.
max_vm_size	196	Максимальный размер виртуальной памяти тестируемой программы.
name	185	Имя тестировщика.
no_core_dump	192	Флаг запрета дампа памяти при фатальных сигналах у тестируемой программы.
no_redirect	190	Флаг отключения перенаправления у тестируемой программы.
prepare_cmd	201	Команда подготовки тестируемой программы.
problem	185	Идентификатор тестируемой задачи.
problem_name	186	Имя тестируемой задачи.
run_dir	198	Каталог обмена программ serve и run.
super	188	Абстрактный тестировщик, свойства которого наследуются.
start_cmd	203	Вспомогательная команда для запуска тестируемой программы.
start_env	204	Переменные окружения для тестируемой программы.
time_limit_adjustment	197	Поправка на время тестирования.

Переменная id

Имя переменной: **id**
Содержится в: tester
Используются: serve, run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *нет*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает идентификатор тестировщика. Идентификатор тестировщика, в отличие от идентификатора задачи или языка программирования, не является ключём при поиске, а используется просто для указания элемента массива тестировщиков, в котором хранится описание данного тестировщика. Поэтому в большинстве случаев явное указание идентификатора тестировщика не требуется. Каждый тестировщик должен иметь уникальный идентификатор в пределах от 1 до MAX_TESTER. Если идентификатор тестировщика явно не задан, он назначается автоматически. Для этого берётся идентификатор предыдущего тестировщика и увеличивается на 1. Данная процедура автоматического назначения тестировщика может приводить к ошибке повторяющегося идентификатора, которая приведёт к ошибке запуска системы.

Абстрактные тестировщики не могут устанавливать переменную id. Если переменная id у абстрактного тестировщика установлена, выдаётся сообщение об ошибке. Идентификатор тестировщика не наследуется от абстрактных тестировщиков неабстрактными тестировщиками.

Пример.

```
id = 10
```

Переменная `name`

Имя переменной: **`name`**
Содержится в: `tester`
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *нет*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт символическое имя тестировщика. Для неабстрактных тестировщиков оно никак не используется. Если имя тестировщика не задано, автоматически устанавливается имя `tst_%d`, если архитектура тестировщика не задана, и `tst_%d_%s`, если архитектура тестировщика задана. Форматное преобразование `%d` применяется к идентификатору тестировщика `id`, а форматное преобразование `%s` применяется к архитектуре тестировщика `arch`.

Для абстрактных тестировщиков имя тестировщика является ключём, по которому находится тестировщик для выполнения операций наследования свойств неабстрактным тестировщиком от данного абстрактного тестировщика. В абстрактных тестировщиках конфигурационная переменная `name` должна быть задана и должна быть уникальна среди всех абстрактных тестировщиков. Естественно, переменная `name` не наследуется.

Пример.

```
name = "Dos_tester"
```

Переменная `problem`

Имя переменной: **`problem`**
Содержится в: `tester`
Используются: serve, run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *нет*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает идентификатор задачи, тестируемой данным тестировщиком. Если значение переменной установлено, задача с таким идентификатором, задаваемым конфигурационной переменной `id`, должна существовать. Абстрактные тестировщики и тестировщики по умолчанию не могут устанавливать данную конфигурационную переменную. Для прочих тестировщиков для идентификации задачи, тестируемой данным тестировщиком, должна быть задана либо конфигурационная переменная `problem`, либо конфигурационная переменная `problem_name`, но не обе одновременно. Конфигурационная переменная `problem` не наследуется.

Пример.

```
problem = 10
```

Переменная problem_name

Имя переменной: **problem_name**
Содержится в: tester
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *нет*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает короткое имя задачи, тестируемой данным тестировщиком. Если значение переменной установлено, задача с таким коротким именем, задаваемым конфигурационной переменной short_name, должна существовать. Абстрактные тестировщики и тестировщики по умолчанию не могут устанавливать данную конфигурационную переменную. Для прочих тестировщиков для идентификации задачи, тестируемой данным тестировщиком, должна быть задана либо конфигурационная переменная problem, либо конфигурационная переменная problem_name, но не обе одновременно. Конфигурационная переменная problem_name не наследуется.

Пример.

```
problem_name = "A"
```

Переменная any

Имя переменной: **any**
Содержится в: tester
Используются: serve, run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *нет*
Значение по умолчанию: false
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, данный тестировщик считается тестировщиком по умолчанию. Такой тестировщик используется для тестирования решений всех задач, для которых не задан явный тестировщик. Тестировщик по умолчанию не может устанавливать конфигурационные переменные problem и problem_name. Тестировщик по умолчанию может иметь абстрактный тестировщик, свойства которого наследуются. Процедура наследования свойств выполняется при тестировании решения задачи, подходящей под данный тестировщик по умолчанию. При наследовании некоторых переменных выполняются форматные подстановки, для которых используются значения конфигурационных переменных описания задачи, соответствующие той неабстрактной задаче, решение которой тестируется. Таким образом, при каждом наследовании свойств значения переменных, зависящие от параметров задачи, будут принимать значения, соответствующие текущей задаче.

Абстрактные тестировщики не могут устанавливать переменную any. Значение этой конфигурационной переменной не наследуется.

Пример.

```
problem_name = "A"
```

Переменная `abstract`

Имя переменной: **`abstract`**
Содержится в: `tester`
Используются: serve, run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *нет*
Значение по умолчанию: `false`
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, текущее описание тестировщика является описанием абстрактного тестировщика.

Абстрактный тестировщик не имеет идентификатора (то есть не может использовать параметр `id`), не может быть тестировщиком по умолчанию (то есть не может устанавливать параметр `any`), не может устанавливать тестируемую задачу с помощью конфигурационных переменных `problem` или `problem_name`. Абстрактный тестировщик не может наследовать свойства другого абстрактного тестировщика, то есть использование конфигурационной переменной `super` в описании абстрактного тестировщика не допускается.

Описание абстрактного тестировщика должно устанавливать параметр `name`, который используется для идентификации абстрактного тестировщика. Имя абстрактного тестировщика должно быть уникальным среди всех абстрактных тестировщиков.

Описание абстрактного тестировщика устанавливает значения конфигурационных переменных. Неабстрактный тестировщик указывает имя абстрактного тестировщика в конфигурационной переменной `super`, при этом значение некоторой переменной наследуется из абстрактного тестировщика только в том случае, если эта переменная не установлена в самом неабстрактном тестировщике.

Конкретный тестировщик может наследовать следующие конфигурационные переменные из описания абстрактного тестировщика: `arch`, `check_cmd`, `check_dir`, `clear_env`, `errorcode_file`, `error_file`, `is_dos`, `key`, `kill_signal`, `max_data_size`, `max_stack_size`, `max_vm_size`, `no_core_dump`, `no_redirect`, `prepare_cmd`, `run_dir`, `start_cmd`, `start_env`, `time_limit_adjustment`.

При этом при наследовании конфигурационных переменных `check_cmd`, `check_dir`, `errorcode_file`, `error_file`, `prepare_cmd`, `run_dir`, `start_cmd` выполняется форматная подстановка (см. раздел 2.11.2).

Не наследуются следующие конфигурационные переменные абстрактной задачи: `abstract`, `any`, `id`), `name`, `problem`, `problem_name`, `super`.

Пример.

```
abstract
```

Переменная `super`

Имя переменной: **`super`**
Содержится в: `tester`
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *нет*
Значение по умолчанию: не установлено
Может повторяться: *нет*

Описание. Данная конфигурационная переменная определяет имя абстрактного тестировщика, свойства которого наследует данный неабстрактный тестировщик. Значением переменной должно быть имя (см. конфигурационную переменную `name`) абстрактного тестировщика. Если абстрактного тестировщика с таким именем не сущесвует, программы serve и run не запускаются. Неабстрактная задача может наследовать свойства единственной абстрактной задачи. Если конфигурационная переменная `super` не установлена, никакого наследования свойств не происходит.

FIXME. Вроде бы абстрактная задача может наследовать свойства нескольких других абстрактных задач. Надо проверить!

Наследование некоторой конфигурационной переменной заключается в копировании её значения из описания абстрактной задачи в описание неабстрактной задачи. Для некоторых конфигурационных переменных при этом выполняются форматные подстановки. Значение переменной наследуется только тогда, когда значение переменной в неабстрактной задаче неопределено. Если неабстрактная задача устанавливает значение некоторой переменной, всегда используется это значение данной переменной.

```
super = "DOS_Tester"
```

Переменная `arch`

Имя переменной: **`arch`**
Содержится в: `tester`
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт «архитектуру», для которой предназначен данный тестировщик. Архитектура является одним из двух параметров, по которому выбирается тестировщик для тестируемой программы (вторым является идентификатор задачи). Например, значение архитектуры `dos` может соответствовать компиляторам для DOS (например, Borland C 3.1), а неустановленное значение архитектуры (пустая строка) может соответствовать компиляторам для Linux, которые генерируют статический исполняемый модуль (например, GCC). Процедура запуска скомпилированной программы для этих архитектур существенно отличается, поэтому для каждой из них должен использоваться отдельный тестировщик.

Данная конфигурационная переменная соответствует переменной `arch` секции описания языка. Тестировщик с некоторой архитектурой предназначен только для программ, скомпилированных под ту же самую архитектуру. Пустое значение переменной `arch` является значимым, то есть в этом случае тестирощик будет тестировать только программы, скомпилированные компилятором с пустым значением переменной `arch`.

Пример.

```
arch = "dos"
```

Переменная `no_redirect`

Имя переменной: **`no_redirect`**
Содержится в: tester
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, у программы, запускаемой на тестирование данным тестировщиком, не будут перенаправляться стандартные потоки вывода и ошибок. Весь вывод тестируемой программы в эти потоки будет потерян. Если данная переменная не установлена или установлена в *false*, стандартные потоки вывода и ошибок перенаправляются в файлы и затем включаются в протокол тестирования и/или проверяются на соответствие правильному ответу. Стандартный поток ввода в любом случае перенаправляется из устройства `/dev/null`.

Данная переменная предназначена для случая, когда программа запускается не непосредственно, а через какую-нибудь вспомогательную программу (например, эмулятор DOS). В этом случае вывод вспомогательной программы может не представлять интереса и поэтому может игнорироваться.

Пример.

```
no_redirect
```

Переменная `is_dos`

Имя переменной: `is_dos`
Содержится в: tester
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если значение данной конфигурационной переменной равно *true*, программы, тестируемые данным тестировщиком, работают с текстовыми файлами в формате DOS. В таких файлах строки текста завершаются двумя символами `\r`, `\n`. Если значение данной переменной не установлено или установлено в *false*, тестируемые программы работают с текстовыми файлами в формате Unix, в которых строки текста завершаются символом `\n`. Как правило, формат текстового файла, подаваемого на вход тестируемой программе, должен соответствовать требуемому. Например, некоторые стандартные функции языка Borland Pascal могут работать некорректно, если программе на вход подаётся текстовый файл в формате Unix, и наоборот, не все программы, написанные для формата Unix, правильно обрабатывают дополнительный символ `\r` в конце строки.

В случае одновременной поддержки языков программирования с текстовыми файлами и в формате DOS, и в формате Unix, тесты для каждой задачи должны быть записаны в текстовом формате Unix, а для тестировщиков, запускающих программы, требущие DOS-формат, должен быть установлен флаг `is_dos`. В этом случае при копировании тестового файла в рабочий каталог будет выполнено его перекодирование в формат DOS. Обратное перекодирование результатов работы программы из формата DOS в формат Unix не выполняется, поэтому проверяющая программа должна корректно обрабатывать оба текстовых формата.

Пример.

```
is_dos
```

Переменная `key`

Имя переменной: **`key`**
Содержится в: `tester`
Используются: serve, run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: `""` (пустая строка)
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает «ключ» для выбора тестировщиков, обслуживаемых программой run. Программа run поддерживает опцию командной строки `-k` с помощью которой можно указать ключ тестировщика. Тогда из всех тестировщиков, описанных в конфигурационном файле, будут обслуживаться только те, ключ которых совпадает с ключём, заданным в командной строке. В этом случае значение по умолчанию (пустая строка) переменной `key` является значимым, и такой тестировщик будет обслуживаться, только если в командной строке в качестве параметра опции `-k` указана пустая строка. Если опция `-k` не задана, отбор языков по значению переменной `key` не производится, и значение этой переменной может быть произвольным (в частности, переменная может быть не установлена).

Значение данной конфигурационной переменной наследуется из описания абстрактного тестировщика, если конкретный тестировщик указал имя этого абстрактного тестировщика в переменной `super`, конкретный тестировщик не определяет переменную `key`, а абстрактный — определяет.

Пример.

```
key = "langs_1"
```

Переменная `no_core_dump`

Имя переменной: **`no_core_dump`**
Содержится в: `tester`
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в значение *true*, при запуске тестируемой программы запрещается генерация дампа памяти (core dump) при аварийном завершении тестируемой программы.

Переменная kill_signal

Имя переменной: **kill_signal**
Содержится в: tester
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: term
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает имя сигнала, который будет послан тестируемой программе при истечении времени её работы, определяемом с помощью конфигурационной переменной real_time_limit. Поддерживаются следующие имена сигналов:

Имя в конфигурационном файле	Системное имя сигнала
kill	SIGKILL
term	SIGTERM
int	SIGINT

Текущая версия системы ejudge ограничивает время выполнения программы на одном тесте (параметр time_limit) с помощью установки ограничения процессорного времени программы с помощью системной функции setrlimit (см. также команду интерпретатора bash ulimit). Такой подход имеет следующие особенности:

1. По истечении добавленной одной секунды тестируемая программа всегда снимается с выполнения посылкой ей сигнала SIGKILL.

2. Если тестируемая программа не занимает процессор, например, выполняя системный вызов pause, её истраченное процессорное время не увеличивается, поэтому программа может находиться в таком состоянии сколь угодно долго.

Для борьбы с последней особенностью используется ограничение астрономического времени работы программы, задаваемое с помощью конфигурационной переменной real_time_limit. Данный параметр kill_signal влияет на сигнал, посылаемый тестируемой программе по истечении астрономического времени работы, но не влияет на сигнал, посылаемый программе по истечении процессорного времени работы.

Пример.

```
kill_signal = "kill"
```

Переменная `max_stack_size`

Имя переменной: **`max_stack_size`**
Содержится в: `tester`
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает максимальный размер сегмента стека для тестируемой программы. Размер задаётся в байтах (так, 8388608 байт — это 8 Мегабайт). Если размер стека превышен, программа, как правило, получает фатальный сигнал `SIGSEGV` (Segmentation fault) и снимается с выполнения. В этом случае тестирование программы завершается с ошибкой “Runtime error”.

Использование данной конфигурационной переменной имеет смысл, если тестируемая программа запускается непосредственно, а не с помощью эмулятора или интерпретатора, так как в этих случаях данная конфигурационная переменная будет ограничивать размер стека эмулятора или интерпретатора.

Если данная конфигурационная переменная не установлена ни в самом неабстрактном тестировщике, ни в абстрактном тестировщике, свойства которого он наследует, ограничение размера стека при запуске программы явно не устанавливается. В этом случае работает ограничение на размер стека, действительное для программы run в момент её запуска. Ограничение на размер стека можно просмотреть с помощью команды `ulimit -s` командного интерпретатора bash.

Пример.

```
max_stack_size = 8388608
```

Переменная `max_data_size`

Имя переменной: **`max_data_size`**
Содержится в: `tester`
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает максимальный размер сегмента данных для тестируемой программы. Размер задаётся в байтах (так, 8388608 байт — это 8 Мегабайт). К сожалению, современные библиотеки поддержки языка Си выделяют динамическую память таким образом, что ограничение на максимальный размер сегмента данных обходится, поэтому для ограничения максимального размера данных программы приходится использовать конфигурационную переменную `max_vm_size`. Использование данной конфигурационной переменной `max_data_size`, по-видимому, не имеет смысла, и она сохранена для симметрии с `max_stack_size`.

Если данная конфигурационная переменная не установлена ни в самом неабстрактном тестировщике, ни в абстрактном тестировщике, свойства которого он наследует, ограничение размера стека при запуске программы явно не устанавливается. В этом случае работает ограничение на размер стека, действительное для программы run в момент её запуска. Ограничение на размер стека можно просмотреть с помощью команды `ulimit -d` командного интерпретатора bash.

Пример.

```
max_data_size = 8388608
```

Переменная `max_vm_size`

Имя переменной: **`max_vm_size`**
Содержится в: tester
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает максимальный размер виртуальной памяти процесса, исполняющего тестируемую программу. В размере виртуальной памяти учитываются сегмент кода (`.text`) самой программы и всех библиотек, используемых программой, сегмент инициализированных статических данных самой программы и всех используемых библиотек (`.data`), сегмент нулевых статических данных программы и библиотек (`.bss`), динамически растущий сегмент динамической памяти («кучи»), динамически растущий сегмент стека. Размер задаётся в байтах (так, 16777216 байт — это 16 Мегабайт).

Если данная конфигурационная переменная не установлена ни в самом неабстрактном тестировщике, ни в абстрактном тестировщике, свойства которого он наследует, ограничение размера стека при запуске программы явно не устанавливается. В этом случае работает ограничение на размер стека, действительное для программы run в момент её запуска. Ограничение на размер стека можно просмотреть с помощью команды `ulimit -v` командного интерпретатора bash.

Пример.

```
max_vm_size = 16777216
```

Переменная `clear_env`

Имя переменной: **`clear_env`**
Содержится в: tester
Используются: run
Тип содержимого: *boolean*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *false*
Может повторяться: *нет*

Описание. Если данная конфигурационная переменная установлена в *true*, при запуске тестируемой программы на выполнение сбрасываются все переменные окружения, и тестируемая программа получает пустое окружение. После сброса всех переменных можно установить новые переменные окружения с помощью конфигурационной переменной start_env.

Пример.

```
clear_env
```

Переменная time_limit_adjustment

Имя переменной: time_limit_adjustment
Содержится в: tester
Используются: run
Тип содержимого: *integer*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: 0
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает «поправку» к ограничению времени тестирования на одном тесте для всех программ, тестируемых данным тестировщиком. Данная переменная задаёт дополнительное время в секундах и должна быть не меньше нуля. Ограничение времени тестирования на одном тесте определяется как сумма значения переменной time_limit описания задачи и значения данной конфигурационной переменной time_limit_adjustment. Если значение переменной time_limit не установлено, то и переменная time_limit_adjustment не используется.

Данная конфигурационная переменная может оказаться полезной, когда тестируемая программа запускается не непосредственно, а из под эмулятора. Тогда во времени тестирования необходимо учесть время старта самого эмулятора. Например, для dosemu это время составляет примерно 0.5 сек. Секундная точность поправки может казаться недостаточной, но, к сожалению, таково ограничение операционной системы (ограничение на процессорное время задаётся с секундной точностью).

Пример.

```
time_limit_adjustment = 1
```

Переменная `run_dir`

Имя переменной: `run_dir`
Содержится в: tester
Используются: serve
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная задаёт путь к каталогам обмена между программами serve и run. Она используется только программой serve, и позволяет для некоторых тестировщиков переопределять путь к каталогу обмена с программой run, устанавливаемый глобальной переменной run_dir. Каждая программа run может иметь единственный каталог обмена, который она, однако, может разделять с другими программами run. Если одновременно несколько программ run (например, находящихся на разных компьютерах в локальной сети) используют один и тот же каталог обмена, расположенный на сетевом диске, запросы на тестирование будут распределяться между ними случайно. Но для того, чтобы часть запросов на тестирование программой serve отдавались одному процессу (или группе процессов, использующих один и тот же каталог обмена), а часть — другому, может использоваться данная конфигурационная переменная.

BUG. Текущая версия системы ejudge никак не использует данную конфигурационную переменную.

Пример.

```
run_dir = /var/ejudge/run/var/compile
```

Переменная `check_dir`

Имя переменной: **`check_dir`**
Содержится в: `tester`
Используются: run
Тип содержимого: *путь к каталогу*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к рабочему каталогу, в котором тестируется решение участника. В этот каталог копируется исполняемая программа и входные данные для неё, в нём же создаются выходные файлы нулевого размера. Решение участника запускается с данным каталогом установленным в качестве текущего каталога. После каждого запуска этот каталог полностью очищается. Для того, чтобы ограничить максимальный размер выходных файлов, создаваемых программой участника, рекомендуется разместить этот каталог на файловой системе ограниченного размера, отдельной от основных файловых систем. Для этого можно использовать, например, loopback-устройство, указав в качестве флагов монтирования `-o loop`.

Если данная переменная не установлена в описании неабстрактного тестировщика, но абстрактный тестировщик, указанный в переменной `super`, устанавливает данную переменную, используется значение, указанное в абстрактном тестировщике, при этом выполняются форматные подстановки. Если после этого значение данной переменной всё равно неопределено, используется значение глобальной конфигурационной переменной `run_check_dir`.

Пример. Для создания небольшой файловой системы, монтируемой с помощью устройства-петли (loopback), можно воспользоваться следующей последовательностью команд.

- Создаём файл требуемого размера (в нашем случае — 32 Мб).

```
dd if=/dev/zero of=/var/ejudge/image bs=1M count=32
```

- Создаём на нём файловую систему.

```
mke2fs -f /var/ejudge/image
```

- Создаём каталог монтирования.

```
mkdir /var/ejudge/disk
```

- Монтируем новую файловую систему.

```
mount /var/ejudge/image /var/ejudge/disk -o loop
```

- Создаём в нём нужный каталог и устанавливаем его права.

```
mkdir /var/ejudge/disk/work
chown user:user /var/ejudge/disk/work
chmod 755 /var/ejudge/disk/work
```

Здесь `user` — пользователь, из-под которого будет проводиться тестирование решений.

После этого в конфигурационном файле турнира `serve.cfg` можно установить глобальную конфигурационную переменную `run_check_dir` следующим образом:

```
run_check_dir = /var/ejudge/disk/work
```

Переменная errorcode_file

Имя переменной: **errorcode_file**
Содержится в: tester
Используются: run
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает имя файла, в котором после завершения выполнения тестируемой программы находится код завершения программы. В этом случае код завершения процесса программы решения участника игнорируется программой run. Эта переменная может оказаться полезной, когда тестируемая программа участника запускается не непосредственно, а из-под эмулятора, который не обеспечивает передачу в вызвавшую эмулятор программу кода завершения эмулируемой программы. Эмулятор DOS dosemu — это пример такого эмулятора. Естественно, в случае использования файла кода завершения решение участника должно запускаться специальной программой (также работающей под эмулятором), которая обеспечит помещение кода завершения программы в требуемый файл.

Если переменная errorcode_file установлена, файл с таким именем будет размещаться в рабочем каталоге тестирования, задаваемом переменной check_dir. Значение переменной errorcode_file не должно содержать символа '/', то есть должно быть именем файла в текущем каталоге.

Пример.

```
errorcode_file = error.out
```

Переменная error_file

Имя переменной: **error_file**
Содержится в: tester
Используются: run
Тип содержимого: *имя файла*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: error
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает имя файла, в который перенаправляется вывод на стандартный поток ошибок тестируемой программы. После окончания тестирования содержимое этого файла вставляется в протокол тестирования. Переменная не используется (то есть перенаправления не происходит), если конфигурационная переменная no_redirect установлена в значение *true*. Файл перенаправления потока ошибок создаётся в рабочем каталоге тестирования, задаваемом конфигурационной переменной check_dir. Значение переменной error_file не должно содержать символа '/', то есть должно быть именем файла в текущем каталоге.

Переменная `prepare_cmd`

Имя переменной: **`prepare_cmd`**
Содержится в: tester
Используются: run
Тип содержимого: *путь к файлу*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к программе, которая подготавливает выполнение тестируемой программы. В качестве параметра подготавливающей программе передаётся имя исполняемого тестируемого файла (без полного пути). Подготавливающая программа запускается в каталоге временных файлов тестировщика, задаваемом глобальной конфигурационной переменной run_work_dir. Её вывод добавляется в протокол тестирования. Подготавливающая программа должна завершаться с кодом возврата 0, или в противном случае тестирование завершится ошибкой “Check Failed”. Подготавливающая программа запускается один раз перед тестированием программы, а не перед каждым запуском программы на очередном тесте.

Если данная переменная в описании тестировщика неопределена, но абстрактный тестировщик, указанный в переменной super определяет эту переменную, используется значение из абстрактного тестировщика, при этом выполняются форматные подстановки. Неопределённое значение переменной допускается, и в этом случае подготавливающая программа не запускается. Если значение данной переменной не начинается с `'/'`, то есть является относительным путём, оно добавляется к значению глобальной конфигурационной переменной script_dir.

FIXME. Нужно, чтобы в программу подготовки передавались бы более вразумительные параметры. Текущий каталог должен устанавливаться, по-видимому, в каталог тестирования check_dir.

Пример.

```
prepare_cmd = "set_perms"
```

Переменная check_cmd

Имя переменной: check_cmd
Содержится в: tester
Используются: run
Тип содержимого: *путь к файлу*
Может отсутствовать: *нет*
Наследуется: *да*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная устанавливает путь к проверяющей программе, которая запускается каждый раз после завершения работы тестируемой программы на очередном тесте и проверяет правильность ответа тестируемой программы. Проверяющая программа запускается только если тестируемая программа уложилась в отведённое на работу над одним тестом время (см. переменную time_limit) и завершилась успешно, то есть выработав код возврата 0. Проверяющей программе передаются два или три параметра в зависимости от того, установлена ли переменная use_corr описания задачи, то есть используется ли при тестировании решения заранее заготовленный файл с правильным ответом.

- Первый параметр (argv[1]) проверяющей программы — путь к файлу, в котором находятся входные данные текущего теста. Передаётся путь к файлу, находящемуся в тестовом каталоге (см. переменную test_dir), а не к файлу, скопированному в каталог тестирования перед запуском тестируемой программы. Таким образом, вне зависимости от значения переменной is_dos проверяющая программа работает с непреобразованным входным файлом.

- Второй параметр (argv[2]) проверяющей программы — путь к файлу, в котором находится результат работы тестируемой программы. Преобразования формата файла не выполняется, поэтому проверяющая программа должна обрабатывать концы строк в и стиле DOS ('\r', '\n') для проверки решения DOS- или win32-программ, и в стиле Unix ('\n') для проверки решения Linux-программ.

- Если установлена конфигурационная переменная use_corr, третий параметр (argv[3]) проверяющей программы — путь к файлу, содержащему правильный ответ на текущий тест (см. переменную corr_dir). Если конфигурационная переменная use_corr не установлена, третий параметр в проверяющую программу не передаётся.

Проверяющая программа запускается с текущим каталогом, установленным в рабочий каталог тестирования. Вывод проверяющей программы на стандартный поток вывода или стандартный поток ошибок сохраняется и добавляется в файл протокола. Результат тестирования определяется по коду возврата, выработанному проверяющей программой.

0	OK — тестируемая программа выдала верный ответ.
4	“Presentation error”
5	“Wrong answer”

Если проверяющая программа выработала любой другой код завершения, завершилась из-за прихода сигнала (любого), или превысила отведённый ей лимит времени (см. переменную checker_real_time_limit), тестирование программы участника завершается со статусом “Check failed”.

Если данная конфигурационная переменная не установлена, но абстрактный тестировщик, указанный в переменной super, устанавливает эту переменную, используется значение переменной из описания абстрактного тестировщика, при этом выполняются форматные подстановки. Если после этого значение данной переменной check_cmd всё ещё неопределено, программа run выдаёт ошибку и отказывается запускаться. Если значение переменной check_cmd не начинается с символа '/', то есть является относительным путём, оно добавляется к значению глобальной конфигурационной переменной checker_dir.

Пример.

```
check_cmd = "check_a"
```

Переменная start_cmd

Имя переменной:	**start_cmd**
Содержится в:	tester
Используются:	run
Тип содержимого:	*путь к файлу*
Может отсутствовать:	*да*
Наследуется:	*да*
Значение по умолчанию:	*не установлено*
Может повторяться:	*нет*

Описание. Данная конфигурационная переменная устанавливает путь к вспомогательной программе, используемой для запуска тестируемой программы участника. Эта вспомогательная программа должна выполнить необходимые подготовительные операции, а затем вызвать тестируемую программу с помощью системного вызова execv или execve. Полный путь к тестируемой программе передается первым аргументом командной строки, а дополнительные аргументы для тестируемой программы — последующими параметрами. Простейший способ вызова тестируемой программы из вспомогательной программы заключается в вызове функции execv со следующими параметрами.

```
execv(argv[1], argv + 1);
```

Идентификатор процесса тестируемой программы должен быть равен идентификатору процесса вспомогательной программы (то есть не допускается использование fork, system, popen и т. д.), так как в противном случае не гарантируется корректное завершение тестируемой программы по истечению лимита времени.

Если данная конфигурационная переменная не установлена, но абстрактный тестировщик, указанный в переменной super, устанавливает эту переменную, используется значение переменной из описания абстрактного тестировщика, при этом выполняются форматные подстановки. Если значение переменной start_cmd не начинается с символа '/', то есть является относительным путём, оно добавляется к значению глобальной конфигурационной переменной script_dir. Если после всех указанных выше действий значение переменной start_cmd неопределено, никакая вспомогательная программа запускаться не будет, а программа run будет запускать тестируемую программу непосредственно.

Пример.

```
start_cmd = "capexec"
```

Переменная `start_env`

Имя переменной: **`start_env`**
Содержится в: `tester`
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Наследуется: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *да*

Описание. Данная конфигурационная переменная позволяет задавать переменные окружения, передаваемые в тестируемую программу. Каждая спецификация переменной окружения имеет вид

```
NAME=VALUE
```

С помощью конфигурационной переменной `start_env` можно устанавливать по одной переменной окружения за раз. Чтобы установить несколько переменных окружения, конфигурационная переменная `start_env` должна использоваться требуемое число раз.

Если и абстрактный тестировщик, указанный в переменной `super` для данного тестировщика, и сам тестировщик устанавливают переменные окружения, спецификации переменных окружения объединяются так, что переменные окружения из абстрактного тестировщика идут первыми, а за ними — переменные окружения из неабстрактного тестировщика.

Пример.

```
clear_env
start_env = "LD_BIND_NOW=1"
start_env = "LD_PRELOAD=/usr/lib/ejudge/libdropcaps.so"
```

2.12 Файл описания теста `test.inf`

В данном разделе описывается формат файла описания теста. В нем определяются такие характеристики теста, как аргументы командной строки, передаваемые тестируемой программе, комментарий к тесту, комментарий для участников (подсказка) к тесту.

2.12.1 Использование

Файлы описания тестов используются только в случае, если конфигурационная переменная `use_info` описания задачи установлена в *true*. Для каждой задачи турнира этот флаг может устанавливаться независимо от других задач. Если конфигурационная переменная `use_info` установлена, то для каждого теста, определенного для задачи, должен существовать файл описания теста. Эти файлы располагаются в каталоге, задаваемом конфигурационной переменной `info_dir` описания задачи. Каждый файл описания теста должен называться *NUM*`.`*SFX*, где *NUM* — номер теста, которому он соответствует, напечатанный тремя цифрами, включая ведущие нули. *SFX* — это суффикс файлов описания тестов, который по умолчанию равен `.inf`, но может быть переопределен с помощью конфигурационной переменной `info_sfx`. Например, если для некоторой задачи определены 3 теста и установлен флаг `use_info`, то в каталоге, определяемом переменной `info_dir`, должны находиться файлы `001.inf`, `002.inf`, `003.inf` (при условии, что используется значение переменной `info_sfx` по умолчанию).

Далее для удобства изложения файл описания теста мы будем называть `test.inf`.

2.12.2 Общая структура

Файл `test.inf` представляет собой текстовый файл, аналогичный по структуре файлу `serve.cfg`. Файл имеет построчную структуру. Строки текста в файле, состоящие только из пробельных символов, игнорируются. Содержимое строки от первого в ней символа `#` и до конца строки игнорируется.

Значимая строка в конфигурационном файле имеет следующий формат:

```
NAME = VALUE
```

Здесь `NAME` — это имя конфигурационной переменной. Имя конфигурационной переменной может состоять из латинских заглавных и строчных букв, цифр и символа подчеркивания. Все поддерживаемые конфигурационные переменные описаны ниже. `VALUE` — это значение переменной. Допускается строка вида

```
NAME
```

В этом случае `VALUE` полагается равным `1`.

`VALUE` может состоять из нескольких «слов», аналогично тому, как команда интерпретатора строки состоит из имени команды и её аргументов. К `VALUE` применяется процедура разбиения строки на слова, аналогичная процедуре, реализованной в командном интерпретаторе bash. Пробельные символы в начале и конце значения `VALUE` отбрасываются. Слова в `VALUE` — это последовательности символов отделённые друг от друга произвольным числом пробельных символов. При разбиении на слова учитываются специальные символы `'`, `"` и `\`.

Символ `'` начинает последовательность символов, которая завершается первым символом `'`. В этой последовательности пробельные символы рассматриваются как обычные символы, и символы `"`, `\` не работают как специальные. После чтения такой последовательности символы `'` с концов отбрасываются.

Символ `"` начинает последовательность символов, которая завершается первым символом `"`. В этой последовательности пробельные символы рассматриваются как обычные символы, и символы `'`, `\` не работают как специальные. После чтения такой последовательности символы `"` с концов отбрасываются.

Символ `\` воздействует на следующий за ним символ. Для некоторых символов (пробельные, `'`, `"`) отменяется их специальный смысл. Для других символов (`n`, `r`) они интерпретируются как специальная запись управляющего кода (как в языке Си). Кроме этого можно задавать код символа в восьмеричной или шестнадцатеричной записи (как в языке Си). Полное описание поддерживаемых последовательностей дано в таблице 2.14. Если подходят одновременно несколько вариантов из данной таблицы, предпочтение отдаётся варианту максимальной длины.

`\` X H_1H_2	Задаёт символ с указанным шестнадцатеричным кодом. Здесь X — символ `x` или `X`, H_1, H_2 — шестнадцатеричные цифры (`[0-9a-fA-F]`).
`\` X H_1	Задаёт символ с указанным шестнадцатеричным кодом. H_1 — шестнадцатеричная цифра.
`\` $O_1O_2O_3$	Задаёт символ с указанным восьмеричным кодом. Здесь O_1 — цифра от `0` до `3`, O_2 и O_3 — восьмеричные цифры (`[0-7]`).
`\` O_1O_2	Задаёт символ с указанным восьмеричным кодом. O_1, O_2 — восьмеричные цифры.
`\` O_1	Задаёт символ с указанным восьмеричным кодом. O_1 — восьмеричная цифра.
`\a`	Символ `\a` в нотации языка Си.
`\b`	Символ `\b` в нотации языка Си.
`\f`	Символ `\f` в нотации языка Си.
`\n`	Символ `\n` в нотации языка Си.
`\r`	Символ `\r` в нотации языка Си.
`\t`	Символ `\t` в нотации языка Си.
`\v`	Символ `\v` в нотации языка Си.
`\` C	Символ C, здесь C — символ, не перечисленный выше.

Таблица 2.14: Действие символа обратной косой черты

Примеры.

Пример 1.

```
␣␣␣␣␣a␣␣b␣␣␣␣c
```

В данном случае строка будет разбита на три слова: `a`, `b` и `c`.

Пример 2.

```
"␣␣a␣"\␣'␣b␣'␣'␣"aa"␣␣'
```

В данном случае строка будет разбита на два слова: `␣␣a␣␣␣b␣` и `␣"aa"␣␣`.

2.12.3 Конфигурационные переменные

В файле описания теста `test.inf` допускаются следующие конфигурационные переменные:

Название	Стр.	Описание
comment	207	Основной (судейский) комментарий к тесту.
params	208	Параметры командной строки для тестируемой программы.
team_comment	207	Комментарий к тесту для участников.

Переменная `comment`

Имя переменной: **`comment`**
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет задать комментарий к тесту. Текст комментария вставляется в судейский протокол тестирования и недоступен участникам турнира. Значение данной конфигурационной переменной должно быть одним словом.

Пример.

```
comment = "Тест на отрицательные X"
```

Переменная `team_comment`

Имя переменной: **`team_comment`**
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет задать комментарий к тесту для участников. Текст этого комментария вставляется в протокол тестирования, доступный команде-автору решения, если на данном тесте решение команды дало неправильный результат. Значение данной конфигурационной переменной должно быть одним словом.

Пример.

```
team_comment = "Проверьте диапазон переменной X!"
```

Переменная `params`

Имя переменной: **`params`**
Используются: run
Тип содержимого: *string*
Может отсутствовать: *да*
Значение по умолчанию: *не установлено*
Может повторяться: *нет*

Описание. Данная конфигурационная переменная позволяет установить параметры командной строки для тестируемой программы. Значение данной переменной может состоять из нескольких слов. Каждое слово будет отдельным параметром командной строки.

Пример.

```
params = a b c
```

В данном случае тестируемой программе будут переданы три параметра командной строки: `a`, `b`, `c`. Как обычно, нулевым параметром командной строки будет передан путь к самой запускаемой программе. Таким образом, общее количество параметров (значение аргумента `argc` в функции `main`) окажется равным 4.