Глава 1

Увод

spoj0 е система за провеждане на състезания по програмиране. Основен фокус пада върху състезания по програмиране в ACM ICPC[1][2] стил, но освен тях може да бъде използван и като помощен инструмент за обучение при курсове свързани с програмирането.

Използването на гъвкави методологии за разработване на софтуер често води до липса на документация. Настоящата дипломна работа се явява и текущо решение на този проблем. Поради подхода за разработване, съдържанието не отговаря на хронологията на разработването, а вместо това представлява сборно описание на анализа, проектирането, реализацията и тестването до момента.

Тестов сървър с работеща инстанция на системата има на http://milo0.no-ip.org/spoj0.

Сорс кодът на системата е достъпен от http://openfmi.net/projects/spoj0/

1.1 Цел

Целта на дипломната работа е създаване на работеща система за състезания и подготовка по програмиране, която да бъде работеща основа за развитието на по-голяма такава с отворен код. Основните цели са минималистичност, простота, отвореност, цялостност, лесно внедряване, лесно използване и лесно разработване. Анализа, проектирането, реализацията и тестването е строго съобразен с целите, като поради това са направени различни компромиси.

Глава 2

Проблемна област

Състезанията по програмиране са основен начин за сравнение на способностите на обучаващите се по информатика (ученици и студенти). Съществуват доста различни по форма и правила състезания, но най-популярните от тях са ACM International Collegiate Programming Contests за студенти и IOI (международна олимпиада по информатика) за ученици.

Към настоящия момент, единствената дефиниция на online judge която може да се намери в уеб пространството е от wikipedia[3]:

Online judge е online система за тестване на програми в състезания по програмиране. Използва се и за подготовка за такива състезания.

Системата може да компилира и изпълнява вашият код тествайки го срещу предварително конструирани данни. Изпратения код може да бъде изпълняван с ограничения, включително ограничено време, ограничена памет, ограничени права и т.н. Изходът от програмата ще бъде прихванат от системата и сравнен с верният изход, след което системата ще върне резултата.

Няки автори[4] предпочитат наименованието "Автоматична оценяваща система"("Automatic Grading System"). Други автори[5][6] предпочитат наименованието Online Judge, което е и по-наложено сред общността на състезателите в България.

2.1 ACM ICPC

ACM International Collegiate Programming Contest[1] (съкращавано като ACM-ICPC или просто ICPC) е ежегодно, състезание в няколко кръга между университетите по света. ICPC предизвиква студентите да поставят по-високи стандарти за отличие сред тях, чрез състезание което

което оценява отборната работа, анализа на проблеми и бързото разработване на софтуер.

2.1.1 История

ACM International Collegiate Programming Contest (ICPC) произхожда от състезание което е проведено в Texas A&M University през 1970-та година. Състезанието еволюира към сегашната си форма на няколко кръга през 1977-ма година, с първите финали проведени заедно с ACM Computer Science Conference.

От 1977 до 1989, състезанието включва главно отбори от САЩ и Канада. Управлявано от Baylor University от 1989, с регионали образувани от световната университетска общност, действащо под покровителството на ACM и със съществена подкрепа от индустрията, ICPC прераства в световно състезание с отбори от 84 държави през 2005.

От поемането на главното спонсорство от IBM през 1997, участниците в състезанието нарастват значително. През 1997 участват 840 отбора от 560 университета, докато през 2006 участват 6099 отбора от 1756 университета. Броят на отборите нараства с 10-20% всяка година и бъдещите състезания могат да бъдат дори по-големи.

ACM ICPC World Finals е финалният кръг на ACM International Collegiate Programming Contest. С развитието на състезанието, той прераства в 4 дневно събитие, провеждано в най-хубавите места по света.

2.1.2 Правила

ACM ICPC е отборно състезание. Текущите правила указват, че всеки отбор се състои от трима студенти. Участниците трябва да бъдат университетски студенти с по-малко от пет години университетско обучение преди състезанието. Студентите които вече са се състезавали в два финални кръга или пет регионални кръга нямат право да се състезават.

По време на състезанието, на отборите се дават 5 часа за да решат между 8 и 12 задачи. Те трябва да изпращат решения като програми на езиците C, C++ или Java. Програмите се стартират върху тестови данни. Ако програмите не успят да произведат верни резултати, отборите се известяват за това и могат да изпратят друга програма.

Победител е отборът който успешно реши най-много задачи. В случай на отбори с равен брой задачи, за класирането се отчита и т.н. наказателно време. Това е сумата от времената в точките в които отборът изпраща вярна задача плюс 20 минути за всеки неуспешен опит на задача която по-късно е решена.

За пример може да разгледаме следната ситуация за отбори Ягодка и Черешка при задачи A, B и C:

- Ягодка изпраща успешно решение на A в 60 минута.
- Черешка изпраща успешно решение на A в 80 минута.
- Ягодка изпраща неуспешно решение на C в 90 минута.
- Черешка изпраща неуспешно решение на C в 100 минута.
- Черешка изпраща успешно решение на C в 120 минута.
- Ягодка изпраща успешно решение на B в 165 минута.

При тази ситуация Ягодка има 2 задачи и 60+165=225 наказателни минути, докато Черешка има две задачи и 80+120+20=220 наказателни минути. Печели Черешка.

Сравнено с други състезания по програмиране като например IOI, ACM ICPC се отличава с големия брой задачи. Друго специфично свойство е, че всеки отбор може да използва само един компютър, въпреки че се състои от трима. Това сериозно повишава напрежението, поради което се изискват сериозни умения за работа в екип, както и умения да се работи под напрежение.

2.1.3 Регионали и финали

Състезанието се провежда на няколко кръга. Много университети провеждат вътрешни състезания за да определят кои студенти ще изпратят на регионалния кръг. След това университетите се състезават в регионално състезание. От всеки регион поне един отбор отива на финалния кръг. По-големи региони изпращат няколко отбора на финалите (понякога до 6 отбора за много големи региони).

Никой състезател не може да се участва в повече от два финални кръга.

Някои големи региони също така провеждат под-регионални състезания, които са междинно ниво между локални и регионални кръгове.

2.1.4 Вътрешни

В България се провеждат голям брой състезания по програмиране. Като част от ACM ICPC могат да се считат:

- Републиканска студентска олимпиада по програмиране.

- Вътрешните състезания в различни университети (включително и ФМИ на СУ).

Републиканската студентска олимпиада по програмиране се провежда всяка година през пролетта. Различни университети са домакини на състезанието, като последното се проведе във Варненският Свободен Университет[7].

По-големите университети като Софийски Университет провеждат вътрешни състезания за да тренират отборите си и за да определят кои отбори да изпратят на между-университетски състезания. В такива университети обикновено има група преподаватели и дори цели курсове които са посветени на подготовката на отборите.

Освен тях, често се организират и провеждат други състезания, някои от които с международно участие.

2.2 Международна олимпиада по информатика

Идеята за провеждането на международна олимпиада по информатика[8] за ученици е била предложена на 24-тата главна конференция на United Nations Educational, Scientific and Cultural Organization (UNESCO) в Париж от българският пратеник Професор Сендов през месец Октомври 1987. Този план е бил включен в петата главна програма на UNESCO. През Май 1989, UNESCO стартира и спонсорира първата Международна Олимпиада по Информатика (IOI).

IOI е една от петте международни научни олимпиада. Главната цел на IOI е да се стимулира интереса в информатиката и информационните технологии. Друга важна цел е да се съберат заедно изключително талантливи ученици от различни страни и да им се даде възможност да обменят научен и културен опит.

IOI се организира ежегодно в и от една от участващите страни. Всяка участваща страна обикновено изпраща четири ученика и двама придружаващи възрастни. Учениците се състезават индивидуално и целят да максимизират точките си, като решават множество от задачи по информатика през двата състезателни дена. Задачите са алгоритмични задачи по програмиране, които трябва да бъдат решение на PC. През останалите дни се организират културни и развлекателни събития.

Места на провеждане[9]:

1. 1989 - Правец, България, 16-19 Май 1989

2. 1990 - Минск, Република Беларус, Съветски Съюз, 15-21 Юли 1990
3. 1991 - Атина, Гърция, 19-25 Май 1991
4. 1992 - Бон, Германия, 11-21 Юли 1992
5. 1993 - Мендоса, Аржентина, 16-25 Октомври 1993
6. 1994 - Ханинге, Швеция, 3-10 Юли 1994
7. 1995 - Айндховен, Холандия, 26 Юни - 3 Юли 1995
8. 1996 - Весприм, Унгария, 25 Юли - 2 Август 1996
9. 1997 - Кейп Таун, Южна Африка, 30 Ноември - 7 Декември 1997
10. 1998 - Сетубал, Португалия, 5-12 Септември 1998
11. 1999 - Анталия, Турция, 9-16 Октомври 1999
12. 2000 - Пекин, Китай, 23-30 Септември 2000
13. 2001 - Тампере, Финландия, 14-21 Юли 2001
14. 2002 - Йонг-Ин, Република Корея, 18-25 Август 2002
15. 2003 - Кеноша, Уисконсин, САЩ, 16-23 Август 2003
16. 2004 - Атина, Гърция, 11-18 Септември 2004
17. 2005 - Нови Сад, Полша, 18-25 Август 2005
18. 2006 - Мерида, Юкатан, Мексико, 13-20 Август 2006
19. 2007 ще се проведе в Загреб, Хърватска, Юли 15 - 22, 2007
20. 2008 ще се проведе в Александрия, Египет, 2008
21. 2009 ще се проведе в Пловдив, България, 2009
22. 2010 ще се проведе в Ватерло, Онтарио, Канада, 2010

По IOI правила се провеждат и редица национални състезания и регионални състезания. В България това са Зимните Математически Празници, Пролетен Турнир по Информатика, както и националната олимпиада по информатика която се провежда в три кръга и определя националния отбор за IOI.

2.4.3 spoj.pl

Проектът SPOJ [17] се разработва от Sphere Interest Group към Факултата по Електроника, Телекомуникации и Информатика в Gdansk University of Technology, под покровителството на катедрата по Алгоритми и моделиране на системи.

Главните части на системата са:

- сигурна (дано) online judge програма проверяваща коректността на алгоритмични програми писани на разнообразни езици за програмиране,
- интуитивна content management система за даващите задачи и администраторите,
- мрежов потребителски интерфейс за съдии, включващ PHP frontend и бази данни с информации за потребители, задачи, изпратени решения и други,
- подсистема за организиране на турнири и състезания (в момента се подготвя).

Струва си и да се отбележи информативната стойност на Sphere Online Judge, който използва и до определена степен дава на потребителите достъп до следните ресурси:

- множество задачи, което съдържа не само избрани задачи от национални и международни състезания по програмиране, но също така и оригинални задачи, внимателно подбрани в взаимодействие с екип от експертни автори асоциирани към проекта
- система даваща online статус, съдържащ статистики и класирания на най-добрите, най-активните и други.

2.4.4 Vallidolid

Vallidolid Online Judge е един от най-старите и действащи online judge сървъри. Неговите услуги са използвани не само за подготовката на различни състезатели, но и от различни автори на книги от областта между които и известната "Programming Challenges"[2]. Към момента системата съдържа над 2000 задачи, над 5 милиона изпратени решения [18] и над 4000 посетителя дневно [19].

Въпреки че използването на системата е свободно, разработването и е затворено, поради което тя не може да се използва за вътрешни състезания.

Глава 3

Анализ

В тази глава предлагаме анализ на системата. Поради итеративния подход по който тя е разработвана, това не е начален анализ който би се дефинирал в началото на разработването, а по-скоро сбор на изискванията които са се дефинирали и реализирали през различните итерации.

За да не се разпръсква информацията прекалено много, в тази глава е включено и грубото проектиране (High Level Design) на системата.

3.1 Потребителска група

Потенциалните потребители на `spoj0` могат да се разделят на следните групи:

- състезатели
- треньори
- администратори
- разработчици

Състезателите са тези които участват в състезания провеждани на системата, или се подготвят върху теми сложени на системата. За щастие тези потребители имат поне минимални познания по програмиране, компилиране и т.н. и се очаква да имат добра компютърна грамотност. Съобразено с това, трябва да проектираме системата така, че да е удобна на т.н. advanced потребители, тъй като почти всички състезатели са такива. Разбира се е хубаво да бъде използваема и от обикновени потребители.

Треньорите са тези които дават задачи/теми/състезания и съблюдават системата, както и се намесват в случай когато е необходимо (например аномалия). При една online judge система често (поне в близко бъдеще) ще се случват непредвидени ситуации, при които е необходима намеса от човек. Треньорите обикновено са бивши състезатели и хора с много опит в програмирането и софтуерните системи, така че те също са в категорията advanced потребители.

Администраторите са тези които инсталират и настройват такава система. Много често те са и треньори, но има случаи в които администраторите просто разбират от внедряване и конфигуриране на софтуер и операционни системи.

Разработчиците са програмисти които добавят (или променят) функционалности по системата. Разглеждаме ги като потребители, поради факта че една online judge система винаги може да се нуждае от определено нагаждане за конкретен случай. До момента не е направена универсална система за състезания, най-вече поради факта, че изискванията за такива системи непрекъснато се развиват. Би трябвало лесно да може да се добави нов вид състезание (правила, оценяване и т.н.), нов език или пък елемент от потребителския интерфейс.

Въпреки специалната категория потребители които ще използват системата, е хубаво дейностите да могат да се изпълняват лесно. Това е и една от нашите главни цели.

3.2 Цели

В резюме, приоритетите на `spoj0` са:

- минималистичност
- простота
- отвореност
- цялостност
- лесно внедряване
- лесно използване
- лесно разработване

3.2.1 Минималистичност

Минималистичността е избрана като приоритет, за да може да се приложи итеративен подход и гъвкави методологии като например Extreme Programming. Благодарение на нея, беше реализирана работеща версия на системата за по-малко от седмица – нещо с което много малко конкурентни решения могат да се похвалят. В по-нататъшното развитие изискванията се приоритизират, като на всяка итерация се реализират само най-необходимите за момента. Тази тенденция се очаква да продължи и за в бъдеще.

3.2.2 Простота

Системата е проектирана така, че да бъде максимално опростена. Това дава добри резултати както за разработване, така и за използване. Допълнително разслояване на системата би се реализирало само ако обемът ѝстане твърде голям за текущата архитектура. Простотата дава възможност лесно да се разбере системата, като по-този начин е по-привлекателна както за потребителите, така и за заинтересованите от допринасяне.

3.2.3 Отвореност

Въпреки че все още по-голямата част от софтуерните проекти са затворени, разработването на проекти с отворен код набира сериозни темпове в последните години. Дори и от гледна точка на печалба, все повече се утвърждава тезата, че от отворени проекти може да се печели [24] [25][26]. Оказва се обаче, че много малка част от съществуващите системи за състезания са със свободен лиценз и отворен код. Имайки предвид некомерсиалната насоченост на повечето от тях, ние смятаме, че разработване на `spoj0` като отворен проект му дава сериозно предимство пред останалите решения.

3.2.4 Цялостност

Въпреки поетапното разработване на системата, не би било допустимо в даден момент, тя да има непълен вид. Независимо какви функционалности се реализират, в края на всяка итерация системата трябва да има цялостен, работещ вид. Поради тази причина, изборът на функционалности за реализация на всяка итерация е особено важен.

3.2.5 Лесно внедряване

В разрез с много свободни проекти, внедряването на `spoj0` е необходимо да бъде лесно. Авторите нямат за цел да печелят от поддръжка, поради което всеки трябва да може да инсталира и конфигурира системата би бил само плюс за използваемостта ѝ. Трудно внедряване би ограничило силно потенциалните потребители на системата.

3.2.6 Лесно използване

Въпреки че потребителите на `spoj0` не са средно-статистически и обикновено са с висока техническа компетентност, произвеждането на неудобна или трудна за използване система би било лошо. Използваемостта на системата трябва да е съобразена с категорията потребители, така че те да могат да работят лесно и бързо.

3.2.7 Лесно разработване

Колкото повече библиотеки и технологии се използват, толкова повече се стеснява кръга от потенциални разработчици. Дори изборът на език за програмиране отрязва потенциалните такива неколкократно. Необходимо е да се минимизират необходимите познания, като в същото време качеството на кода се запази на добро ниво. За съжаление качеството на кода и необходимите познания за програмиране са фактори които обикновено се движат еднопосочно, поради което е необходим добър баланс, имайки предвид, че целим високо качество, но ниски изисквания за уменията.

3.2.8 Антицели

За `spoj0` не са приоритети:

- рядко необходими функционалности
- шарен дизайн, летящи менюта и снежинки (през зимата)
- поддръжка на всички езици на които може да се напише hello world
- поддръжка на всевъзможни операционни системи
- супер качество на кода
- невероятно проектиране използващо всички design pattern-и

Залитането в някоя от изброените посоки, би имало силно негативен ефект върху целите на spoj0. При по-нататъшно развитие е важно да се подбират внимателно изискванията, за да не се превърне в неизползваема, неразработваема или непотребна система.

3.3 Особености около името

Системата има име spoj0 (произнася се "сподж нула" "споджа" или "споджо"). В духа на много Linux проекти, главните и малките букви имат значение, поради което коректното изписване на името е само с малки букви.

Трябва да се отбележи, че първата версия на тази система излезе под името spoj. Така тя тръгна под тестова експлоатация. При анонсирането ѝ обаче, се оказа, че система с такова име вече има, при това с висока популярност. Имайки предвид колко online judge системи има реализирани изобщо, това е доста малко вероятно съвпадение но все пак името трябваше да бъде сменено. След едноседмично обмисляне, името на системата бе сменено на spoj0. Самата дума е абревиатура, на която бяха измислени много значения. Може би най-официално звучащото от тях е "Sofia Public Online Judge 0".

3.4 Лиценз

За разлика от други проекти, spoj0 се разработва под GNU General Public License версия 2. Това го прави свободен за използване, разпространяване и модификация, като в същото време го предпазва от затваряне и комерсиализиране. Пълен текст на лиценза има в приложенията.

3.5 Use cases

В тази секция представяме описание на основните дейности които биха извършвали потребителите.

3.5.1 Състезатели

влизане в системата

В системата се влиза чрез уеб браузър. Поради простотата на системата, тя би трябвало да работи за всички популярни браузъри. Въвеждане

на потребителски данни е необходимо едва при операция която изисква такива.

гледане на текущото време на сървъра

Текущото време (час/дата) на сървъра се показва на всички страници на системата. Това е съществено, тъй като състезанията се провеждат спрямо него (както и оценяването).

регистриране

За да се изпращат решения на задачи от състезания, е необходимо потребителите да са регистрирани. За регистрация се отива на страницата "register" от навигационното меню, където се попълват необходимите данни. Регистрацията е глобална (за всички състезания/теми). Политиката на системата е да не бъдат ограничавани потребители за участие. Възможно е обаче в бъдеще да се даде възможност за ограничаване на потребителите които участват в официалното (online) класиране.

гледане на състезания

За да се види списъкът с теми/състезания се избира "contests" от навигационното меню.

детайлен преглед на състезание

За да се види конкретно състезания, трябва то да се избере от списъкът със състезания. Тогава се показва детайлна информация за него.

четене на условие

За да се прочете условие на задача, трябва да се достигне до нея от детайлния преглед на съответното състезание. Условията не могат да бъдат четени преди състезанието да е започнало.

изпращане на решение

Изпращане на решение може да се достигне от страницата на съответното състезание, или чрез "submit" от навигационното меню, но в този случай е необходимо да се знае идентификаторът на задачата. Въвеждат се необходимите данни включително и сорс кодът на решението и формата се изпраща.

Сорс код може да се качва както от файл, така и чрез копиране на съдържанието му в съответното поле.

гледане на класиране

До класирането за дадено състезание се достига от страницата със състезанията.

гледане на статус на решения

Статус на изпратените решения може да се види от страницата "status" от навигационното меню. Там се показват последните 100 изпратени решения.

гледане на лог на изпълнение на решения

За да се види лог от изпълнението на решение, то трябва да се избере от страницата със статуси на решения. Логът се вижда само ако състезанието е приключило и това е разрешено в настройките на състезанието, както и при грешка при компилация.

гледане на сорс код на решения

За да се види сорс кода на изпратено решение (включително и от друг състезател), то трябва да се избере от страницата със статуси на решения. Той се вижда само ако състезанието е приключило, и това е разрешено в настройките на състезанието.

3.5.2 Треньори

влизане в системата

В системата се влиза чрез някой от начините за влизане в GNU/Linux система с потребителско име и парола. В home директорията на потребителя на системата се намира всичко необходимо за контрол върху нея.

създаване на състезание

За да се създаде състезание е необходимо неговите данни (структурирани по подходящ начин) да се копират в sets директорията на системата. След това се изпълнява следната команда:

```
./spoj0-control.pl import-set <set_code>
```

където <set_code> е кодът на съответната тема/състезание. Ако всичко е наред се изпращат авторските решения (ако ги има) под тестов потребител.

корекция на състезание

Тъй като не може да се очаква, че треньорите са безгрешни е предвидена възможност за корекция на темите. За да се направи това трябва да се коригират данните за състезанието от sets директорията, след което да се изпълни следната команда:

```
./spoj0-control.pl sync-set <set_code>
```

където <set_code> е кодът на съответната тема/състезание. Ако всичко е наред се изпращат авторските решения (ако ги има) от тестов потребител.

препроверяване на всички неприети решения на дадена задача

При откриване на грешка в тестовите данни, ограничението за време или нещо друго, което би ощетило състезатели се налага препроверяване на изпратените решения. Тъй като в повечето състезания (най-вече в правилата на ACM) приета задача не може да бъде върната, най-честото необходимо действие, е препроверяване на всички неприети. За целта трябва да се изпълни следната команда:

```
./spoj0-control.pl rejudge-problem <problem_id>
```

където <problem_id> е идентификаторът на съответната задача.

препроверяване на всички решения на дадена задача

В случай, че противно на приетите практики за състезания е необходимо преоценяване на всички решения, включително и вече приетите (което може да е подходящо например при задачи за домашна работа) се изпълнява следната команда:

```
./spoj0-control.pl rejudge-problem-all <problem_id>
```

където <problem_id> е идентификатор на съответната задача.

препроверяване на конкретно решение

За преоценяване на конкретно изпратено решение се изпълнява:

```
./spoj0-control.pl rejudge-run <run_id>
```

където <run_id> е идентификатор на изпратено решение.

добавяне на новини

За да се добави новина/съобщение в системата то трябва да се запише във файл в news директорията на системата след което да се извика:

```
./spoj0-control.pl sync-news
```

тестово изпращане на решение

Треньорите имат права да изпращат решения по всички задачи от всякакви потребители. Това може да се прави за да се изпробва авторско решение (в който случай се изпраща от някой от тестовите потребители), или в случай, че състезател е възпрепятстван да го направи сам. Тази функционалност може да се използва и за автоматизирано изпращане на решения от скрипт. Трябва да се изпълни следната команда:

```
./spoj0-control.pl submit <problem_id> <user_id>
    <source_file> <language> [<about>]
```

където <problem_id> е идентификатор на задача, <user_id> е идентификатор потребител от който да бъде изпратена задачата, <language> е код на езика на който да бъде проверена (например "cpp"), а <about> е коментар към самото решение (който може и да го не е зададен).

Тази функционалност може да се използва и като мост от други системи към spoj0.

3.5.3 Администратори

Влизане в системата

В системата се влиза чрез някой от начините за влизане в GNU/Linux система с потребител root.

Инсталиране на системата

Инсталирането на системата върху Debian базирана GNU/Linux дистрибуция става чрез следните две команди:

```
svn export https://svn.openfmi.net/spoj0/spoj0-install.sh
./spoj0-install.sh
```

след което се следва инсталиращия скрипт. При други GNU/Linux системи инсталиращият скрипт може да се нуждае от корекции.

Разпределено стартиране

Възможно е (и много често препоръчително) системата да бъде разпределена да работи няколко компютъра. Единствената част която не може да се разпредели е базата данни (освен ако не се използва разпределена база данни, но този подход не е изследван). Най-често нужда от разделяне има тестващият демон. При големи натоварвания, може да е полезно и разпределяне на уеб достъпа. За да се постигне разпределяне:

- Инсталира се `spoj0` на машината на която ще бъде базата данни.
- На останалите машини:
  - Инсталира се `spoj0` но без да се създава база.
  - Модифицира се spoj0-common така, че да се свързва към машината с базата.
  - Ако е необходимо да има тестващ демон се монтира sets директорията като мрежова файлова система към машината с базата. Алтернативен вариант е физическо копиране/синхронизиране, но при чести промени на данните той не е много удачен.
  - Стартира се каквото е необходимо (обикновено тестващия демон или уеб интерфейса)

Въпреки, че до момента не сме изпробвали този подход поради липса на хардуер, цялата система се проектира така, че той да може да се приложи.

стартиране на демона

Изпълнява се:

```
./spoj0-control.pl start
```

Изисква root права. Удачно тази команда да се добави в `/etc/rc.local` или другаде, за да се извиква автоматично при стартиране. Демонът работи докато не бъде спрян с подходяща команда.

стартиране на демона на място

В случай когато липсват root права, или пък за кратко оценяване, е удачно да се стартира на място. Изпълнява се:

```
./spoj0-control.pl start-here
```

спиране на демона

За да се сигнализира на демона, че трябва да спре се изпълнява:

```
./spoj0-control.pl stop
```

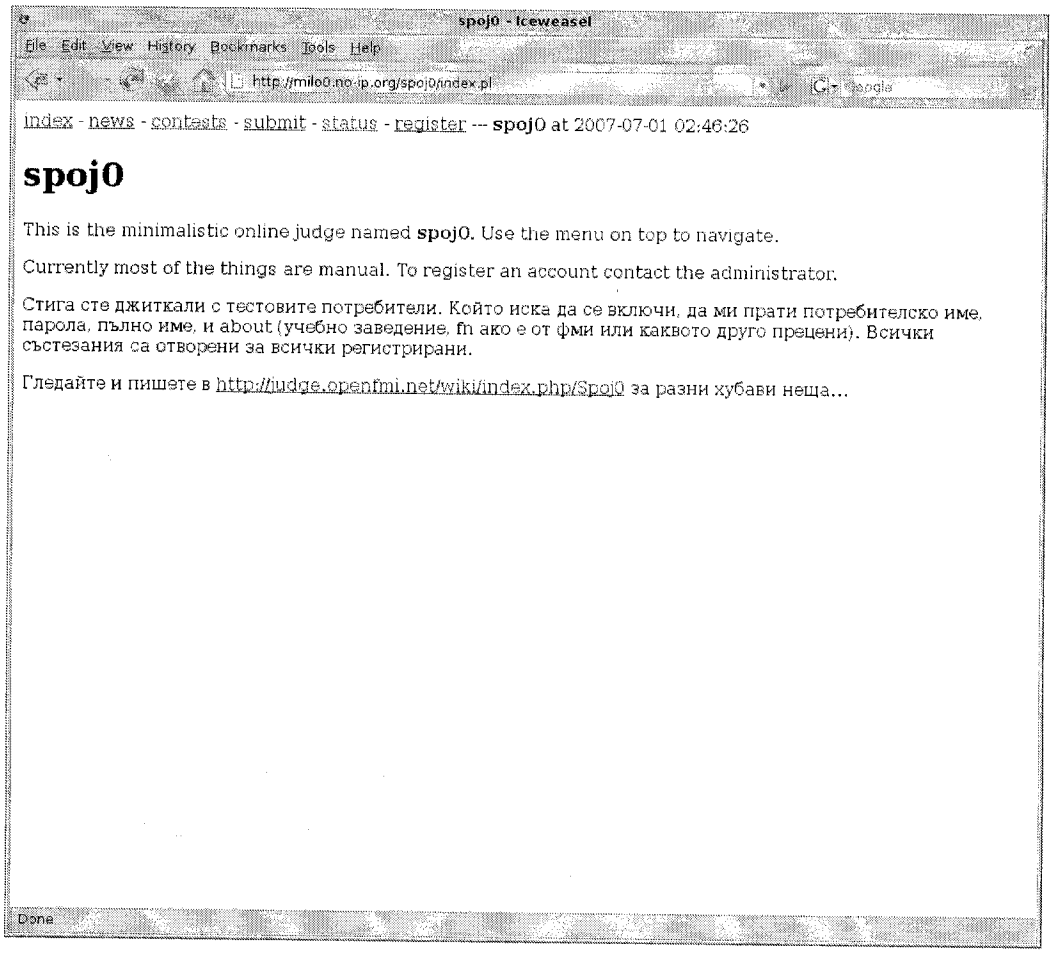
Тази команда е безопасна, за употреба и не би оставила частично проверени решения.

убиване на демона в случай на блокиране

В случай на зависване (което не сме наблюдавали до сега) от демона, може да се изпълни:

```
./spoj0-control.pl kill
```

Тази команда е опасна, тъй като би могла да остави решение в "judging" състояние, което да трябва ръчно да се препровери. Трябва да се използва само в краен случай.

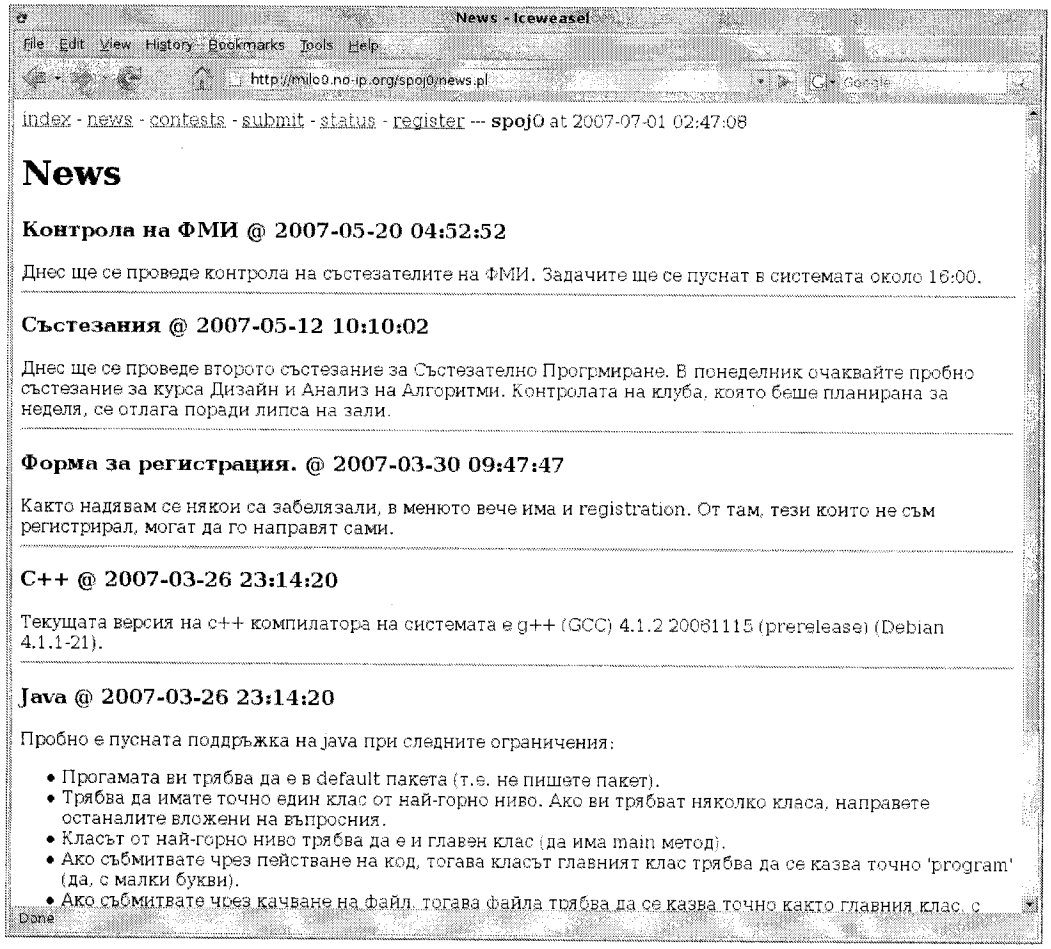


Фигура 4.9: Изглед на index.pl

- коментар
- начален дата/час
- продължителност
- хипервръзка към класирането
- хипервръзка към самото състезание/тема

submit

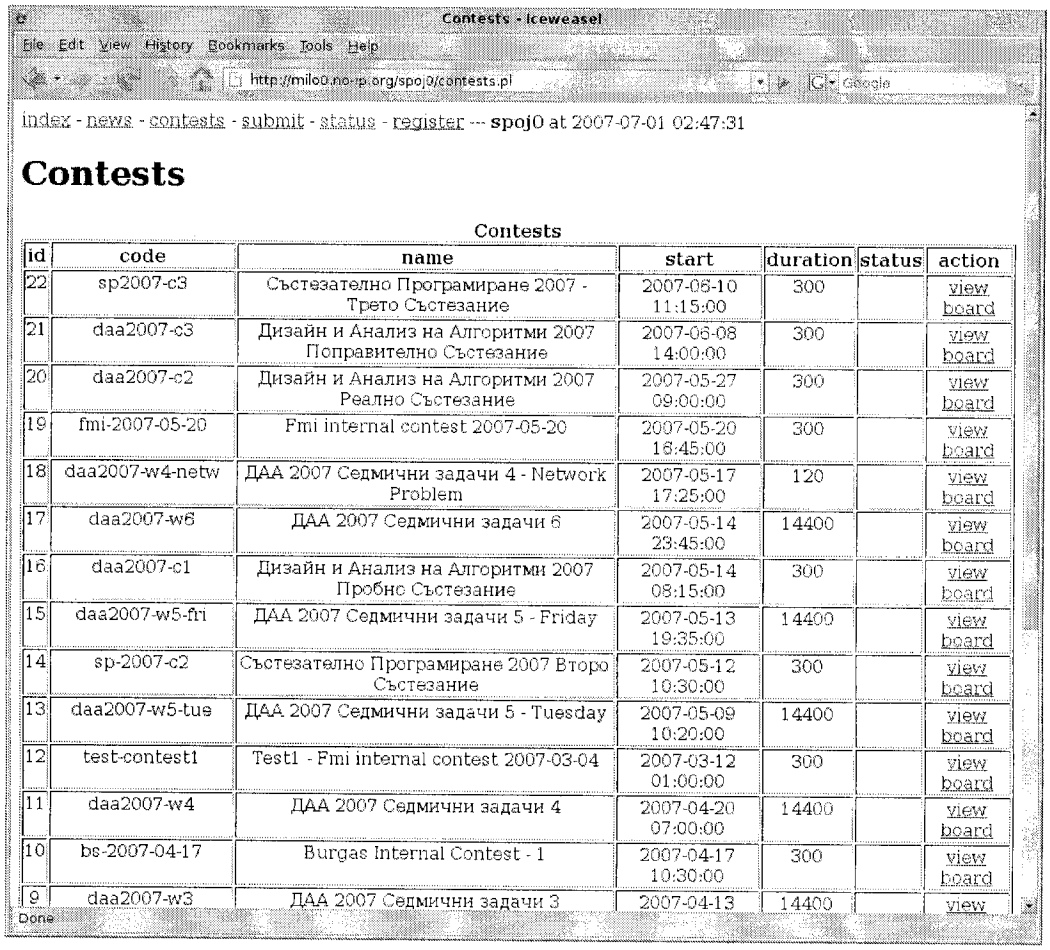
Страница за изпращане на решения. Тъй като уеб интерфейса е stateless при всяко изпращане се въвежда потребителско име/парола. Това не е



Фигура 4.10: Изглед на news.pl

особено проблем от към използваемост, тъй като самата страница запазва стойностите им. Информацията която се въвежда за изпращане на решение е:

- потребителско име
- парола
- идентификатор на проблем (който е попълнен автоматично ако страницата е достъпена през подходящата препратка)
- коментар към решението (незадължителен)
- език за програмиране (към момента C++ или Java)
- поле за качване на файл



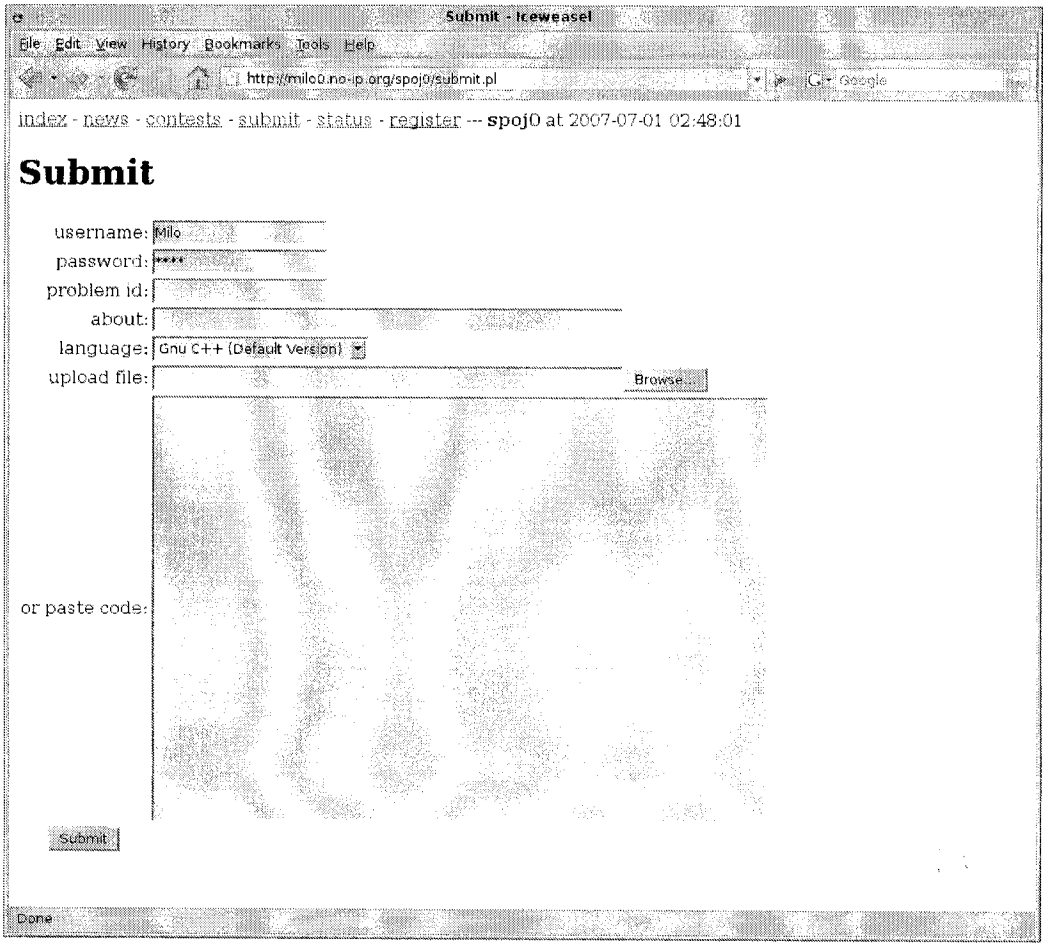
Фигура 4.11: Изглед на contests.pl

- текстова област за програмен код

Възможно е кода на програмата да се качи като файл, или съдържанието му да се копира в текстовата област за това. При вторият вариант обаче, се губи името на файла от който е дошло, което е важно за Java. Поради това, засега има ограничение, ако се изпращат Java решения по втория начин, главния клас да се казва program. В бъдеще е възможно да се реализира автоматично засичане на главния клас и съответно това ограничение да бъде премахнато.

status

Показва опашката с изпратените решения и състоянието на всяко от тях. За всяко изпратено решение показва хипервръзка към детайлния му преглед.



Фигура 4.12: Изглед на submit.pl

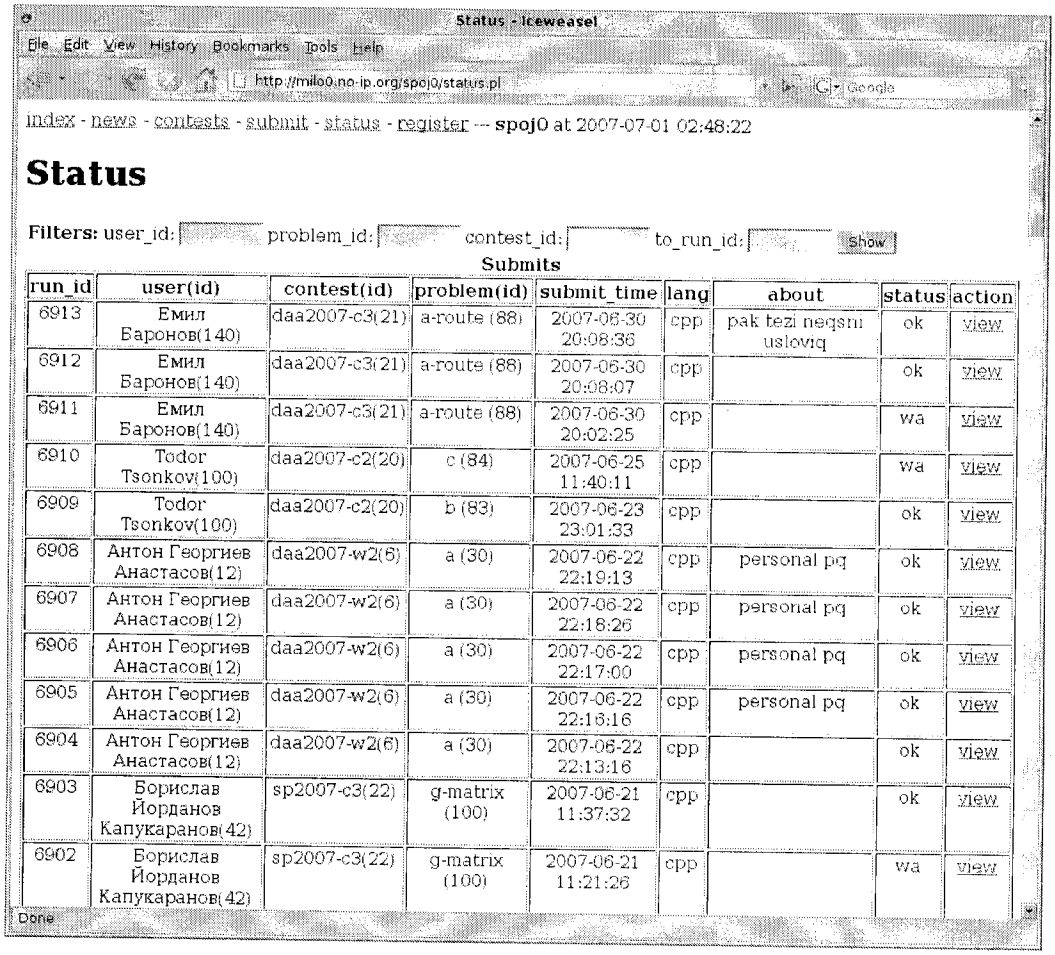
Реализирани са и филтрите за тази страница. Те трябва да предлагат възможност за показване на изпратените решения само по:

- определен потребител
- определено състезание
- определена задача

Към момента, от гледна точка на производителността и натоварването на сървъра се показват най-много последните 100 решения.

status (run_id)

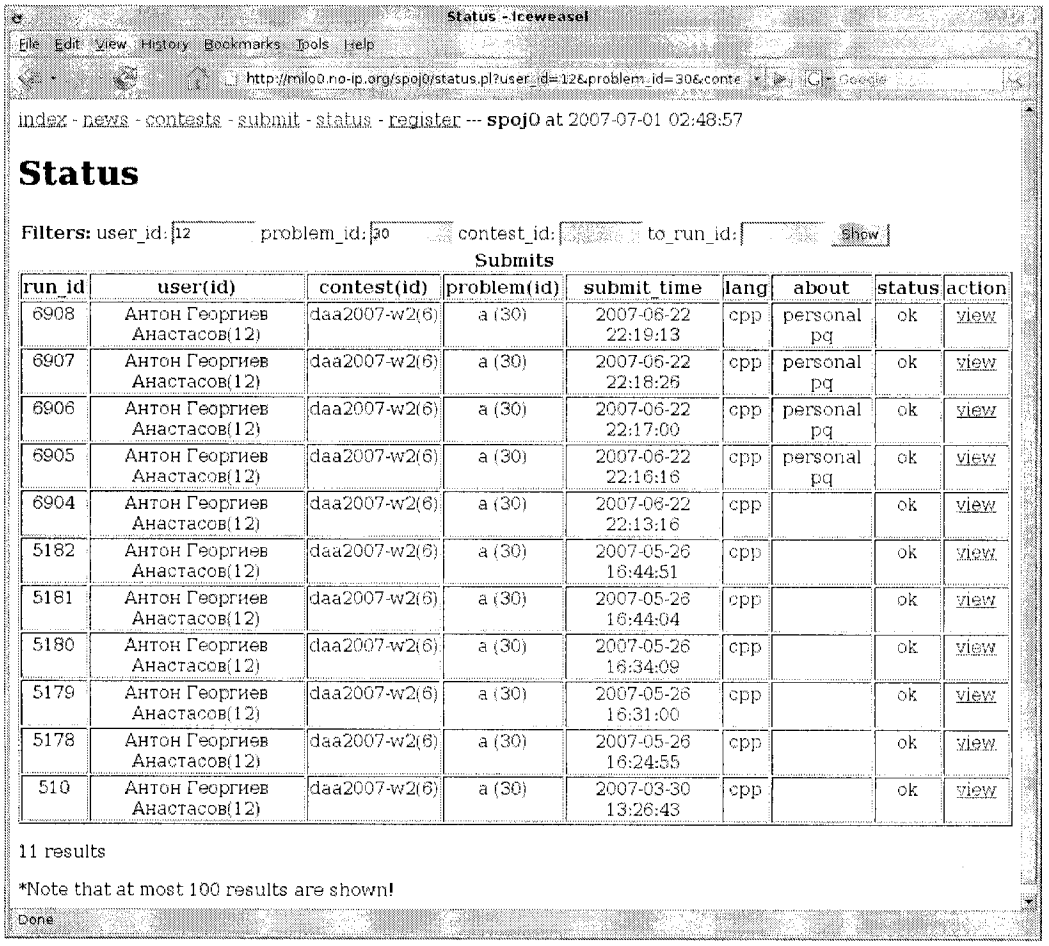
Детайлен преглед на дадено изпратено решение.
Показва следната информация:



run_id	user(id)	contest(id)	problem(id)	submit_time	lang	about	status	action
6913	Емил Баронов(140)	daa2007-c3(21)	a-route (88)	2007-06-30 20:08:36	cpp	pak tezi neqsni usloviq	ok	view
6912	Емил Баронов(140)	daa2007-c3(21)	a-route (88)	2007-06-30 20:08:07	cpp		ok	view
6911	Емил Баронов(140)	daa2007-c3(21)	a-route (88)	2007-06-30 20:02:25	cpp		wa	view
6910	Todor Tsonkov(100)	daa2007-c2(20)	c (84)	2007-06-25 11:40:11	cpp		wa	view
6909	Todor Tsonkov(100)	daa2007-c2(20)	b (83)	2007-06-23 23:01:33	cpp		ok	view
6908	Антон Георгиев Анастасов(12)	daa2007-w2(6)	a (30)	2007-06-22 22:19:13	cpp	personal pq	ok	view
6907	Антон Георгиев Анастасов(12)	daa2007-w2(6)	a (30)	2007-06-22 22:18:26	cpp	personal pq	ok	view
6906	Антон Георгиев Анастасов(12)	daa2007-w2(6)	a (30)	2007-06-22 22:17:00	cpp	personal pq	ok	view
6905	Антон Георгиев Анастасов(12)	daa2007-w2(6)	a (30)	2007-06-22 22:16:16	cpp	personal pq	ok	view
6904	Антон Георгиев Анастасов(12)	daa2007-w2(6)	a (30)	2007-06-22 22:13:16	cpp		ok	view
6903	Борислав Йорданов Капукаранов(42)	sp2007-c3(22)	g-matrix (100)	2007-06-21 11:37:32	cpp		ok	view
6902	Борислав Йорданов Капукаранов(42)	sp2007-c3(22)	g-matrix (100)	2007-06-21 11:21:26	cpp		wa	view

Фигура 4.13: Изглед на status.pl

- идентификатор на изпратеното решение
- дата/час на изпращане
- коментар
- статус
- потребител който го е изпратил
- състезание
- проблем
- *пълен сорс код



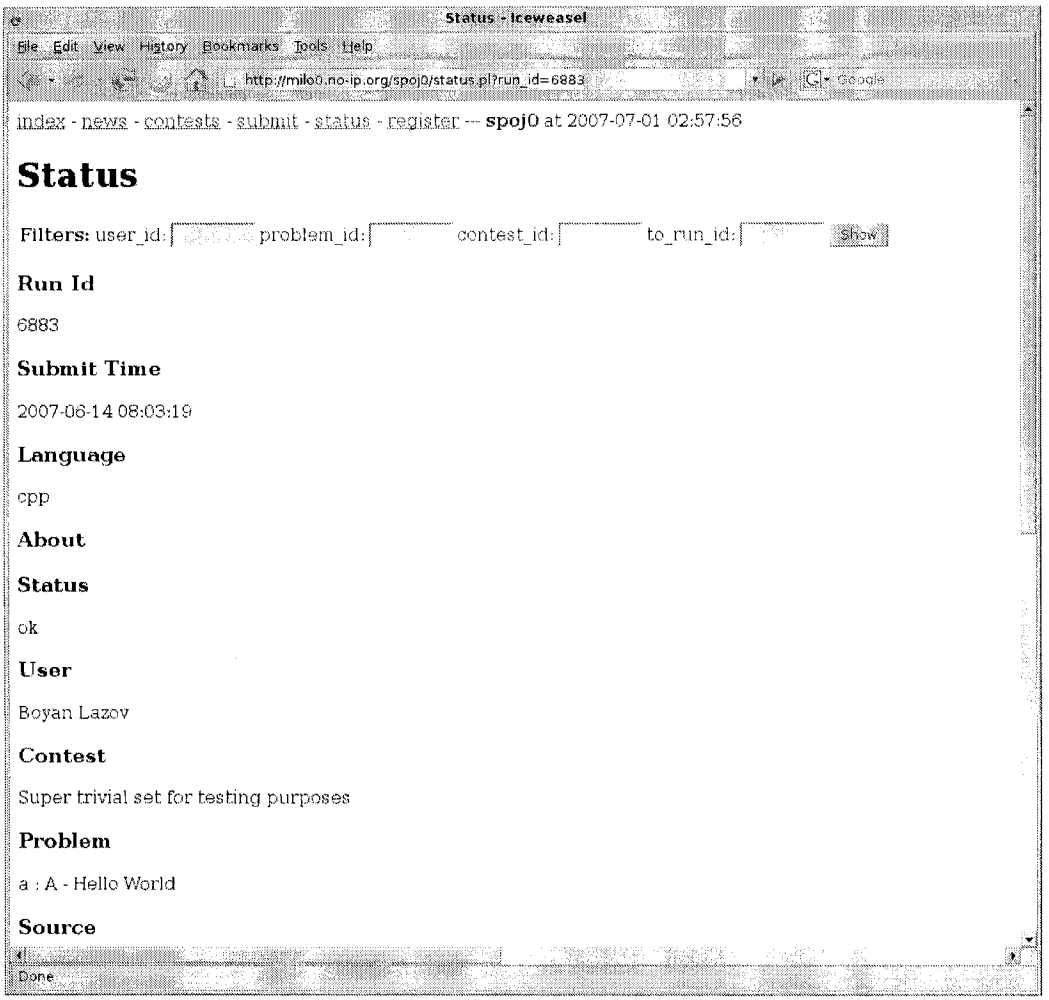
Фигура 4.14: Изглед на status.pl с филтри

- *пълен лог от изпълнение

Сорс кодът се показва само ако състезанието е приключило и съответният флаг в настройките на състезанието казва той да бъде показан.

Лог файлът се показва само ако състезанието е приключило и съответният флаг в настройките на състезанието казва той да бъде показан, както и ако решението е дало грешка при компилация. Последното е важно, тъй като някои потребители не разполагат с идентичен като на системата компилатор и е необходимо да могат да видят къде е проблема.

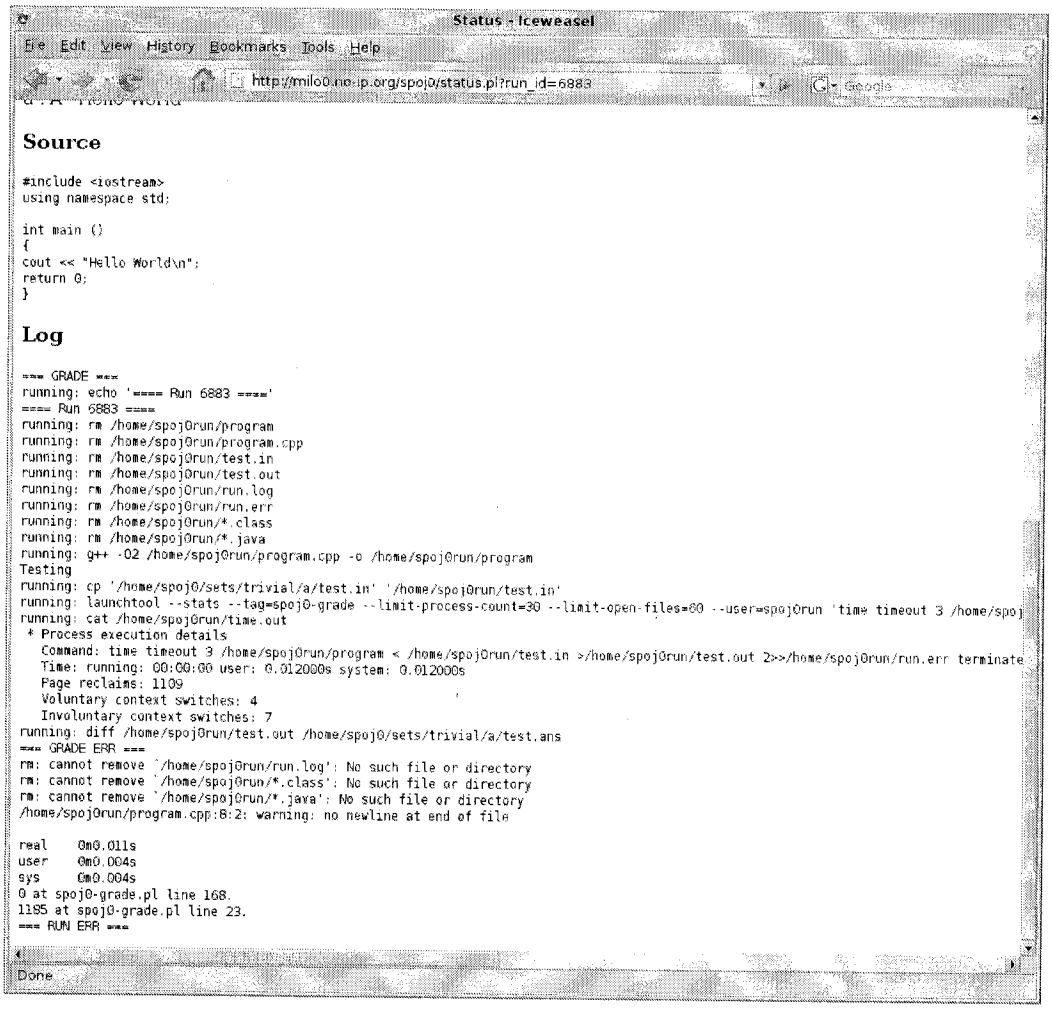
Както отбелязахме по-горе, смятаме, че възможността за показване на сорс кода и лога имат сериозен принос за подобряване на ефекта при използване на системата за учебни цели или подготовка за състезания.



Фигура 4.15: Изглед на status.pl за дадено решение

register

Форма за регистрация на потребител. Дава възможност на потребители да се регистрират сами, като изберат потребителско име, парола и допълнителната информация за тях. Първоначално беше предвидено, от съображения за сигурност потребителите да се регистрират само от треньор или администратор. По-късно обаче се оказа, че трябва да се регистрират доста по-голям брой потребители в системата и това би отнело много време ако се прави само от един човек. Към текущият момент в тестовия сървър на системата има над 120 реални потребители.

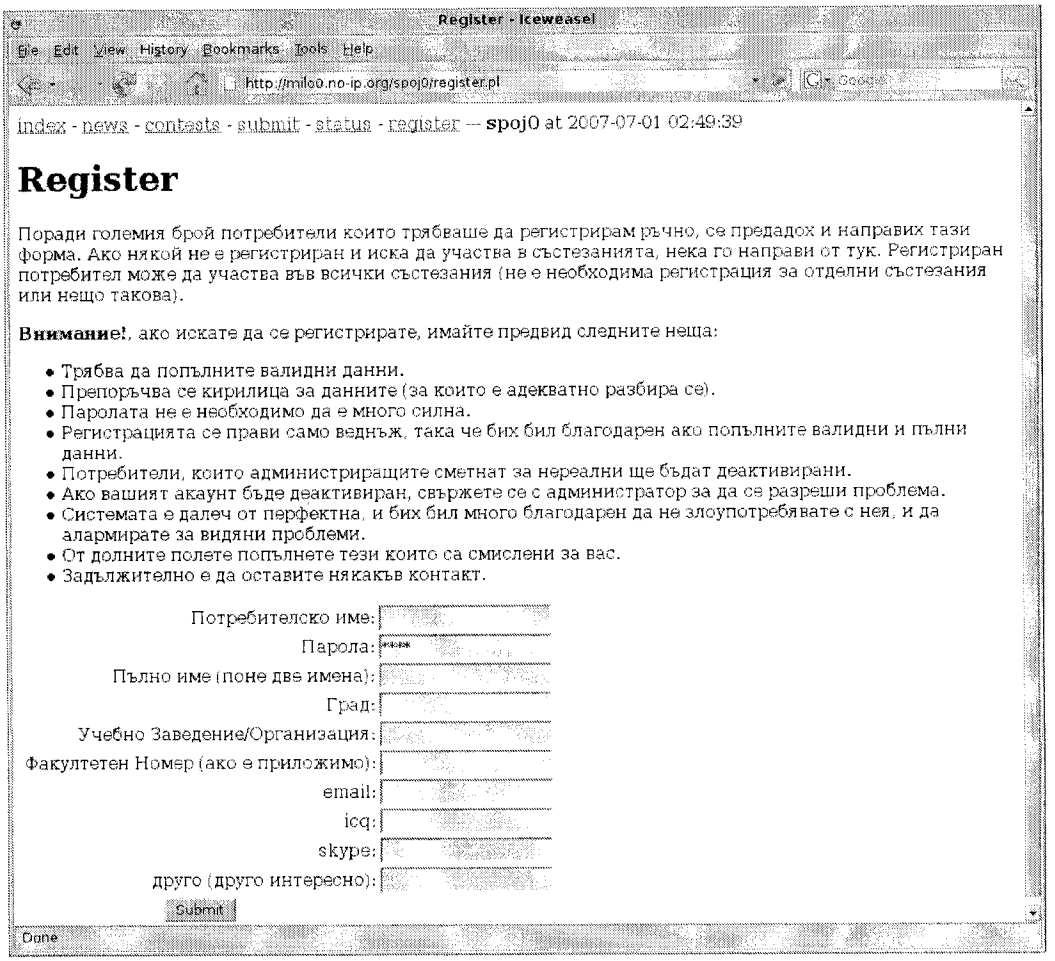


Фигура 4.16: Изглед на кодът на дадено решение от status.pl

description (problem_id)

Показва условие на дадена задача. Към момента просто дава на потребителя да свали условието. Това дава възможност да се поддържат произволни формати за условия, стига условията да са в един файл. Възможно е (и много популярно) всички условия от дадена тема да бъдат в един голям файл, като в този случай се дава за сваляне него. Въпреки, че в системата е предвидено да се подържат повече от един формат на едно и също условие, към момента уеб интерфейса дава за сваляне само първият.

Ако състезанието от което е съответният проблем още не е започнало, се показва подходяща грешка.

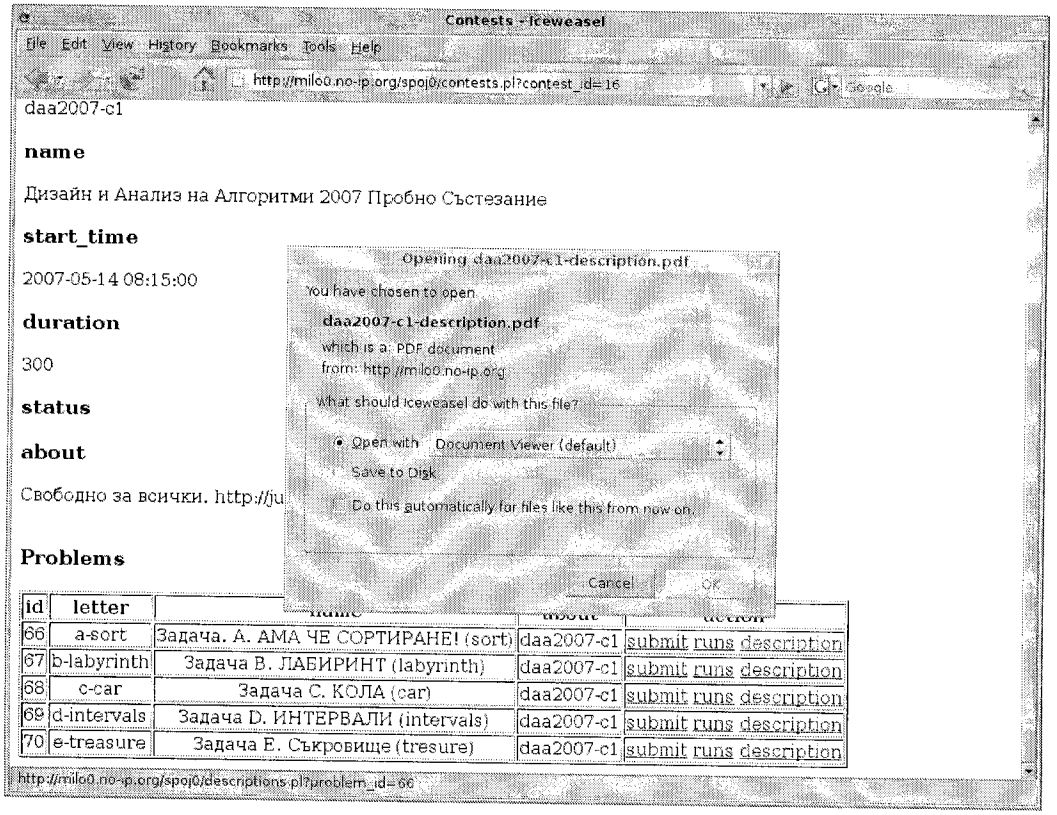


Фигура 4.17: Изглед на register.pl

board (contest_id)

Показва online и offline класиране за съответното състезание в ACM стил. В класирането участват всички потребители изпращали решения по съответното състезание.

Online класирането се формира по правилата на ACM ICPC, като в него участват решенията изпратени по време на самото състезание. Подобно класиране има във всички системи които поддържат ACM състезания. Offline класирането обаче е функционалност специфична за `spoj0` (поне към текущия момент). При нея участват всички решения изпратени след началото на състезанието, включително и след края му. Политика при `spoj0` е задачите да могат да се изпращат и след края на състезанието за да може да се прилага практиката дорешаване. Offline класирането включва online класирането, но показва и резултатите от то-



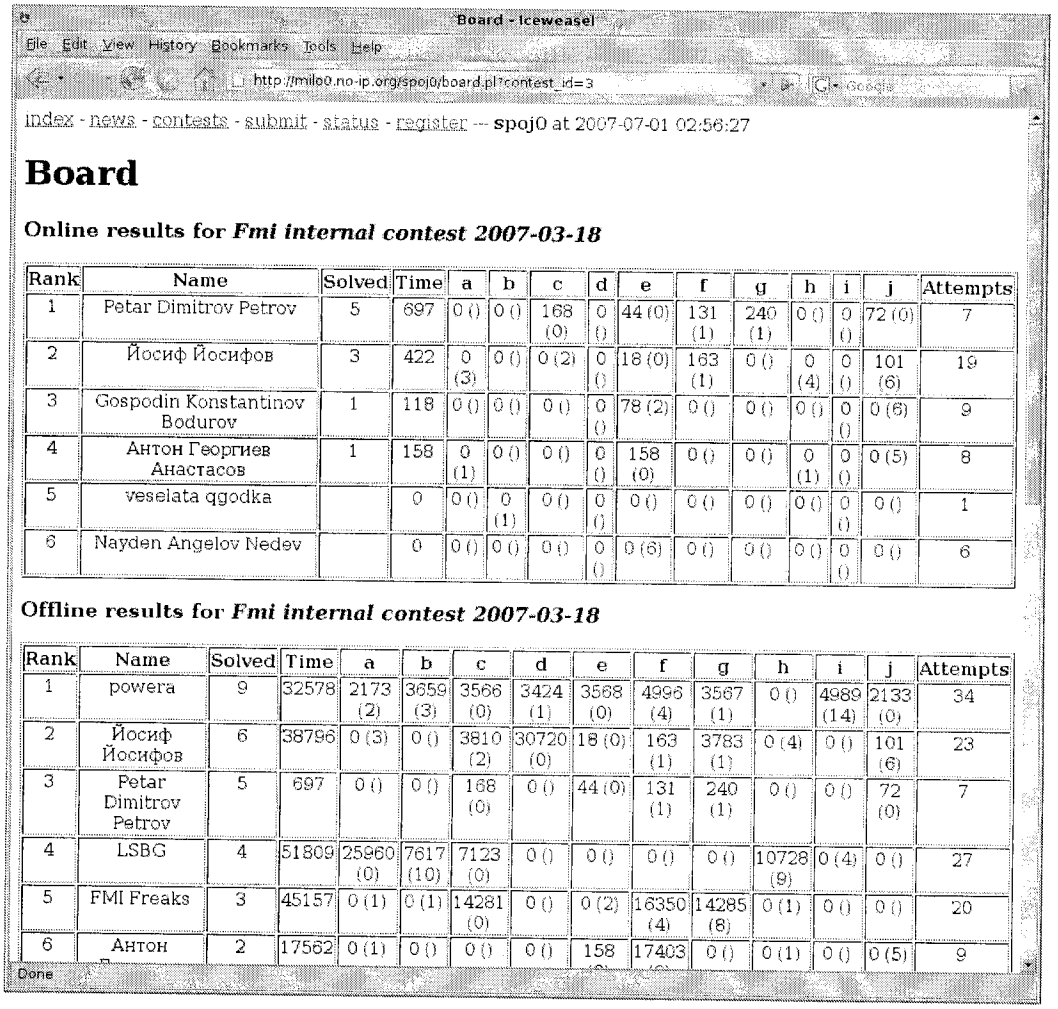
Фигура 4.18: Изглед на description.pl

ва дорешаване. Разбира се сумарните времена при него могат да бъдат доста големи, тъй като ограничение на времето за изпращане няма.

contests (contest_id)

Показва детайлна информация за дадено състезание/тема. Това включва:

- идентификатор
- код
- име
- начална дата/час
- продължителност
- коментар



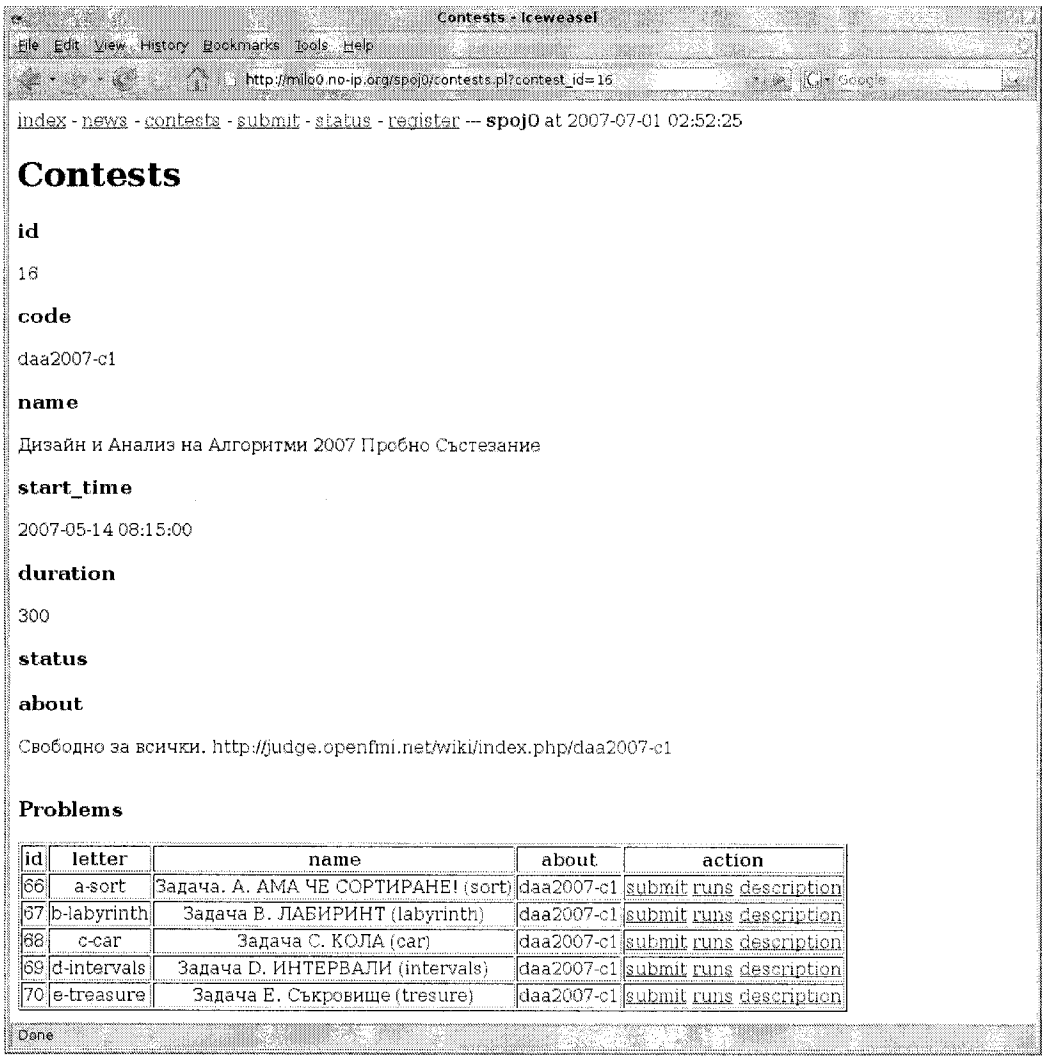
Фигура 4.19: Изглед на board.pl

- списък със задачите в него

За всяка от задачите в състезанието има:

- идентификатор
- буква/код
- име
- коментар
- хипервръзка за изпращане на решение
- хипервръзка към статус на изпратените решения

- хипервръзка към условието



Фигура 4.20: Изглед на contests.pl за конкретно състезание

4.5.5 spoj0-daemon

Тестващият демон е програма която работи непрекъснато на проверяващите машини. Тя е проектирана да може да работи в background режим и се пуска или спира от администратор. Нейната работа е периодично да намира изпратено решение което трябва да бъде проверено, но не е и да извиква spoj0-grade за него. При проектирането и реализацията

са взети предвид възможността на конкурентна работа на няколко демона. Обикновено няма полза да се пускат повече от един демон на една машина, но това все пак е възможно, тъй като противно ограничение би било изкуствено.

Безопасната конкурентна работа на демона е реализирана като се разчита атомарността (транзакционната безопасност) на единичните заявки към базата. В случай на неуспешно оценяване от страна на spoj0-grade (т.е. вътрешна/системна грешка) съответните решение заспива в judging състояние, но демона продължава да работи и да оценява други. При такава ситуация (която обикновено означава бъг или неконфигурираност на системата) се очаква намеса от страна на треньор или администратор за разрешаване на проблема.

4.5.6 spoj0-grade

Оценителят е програма на Perl която приема само един аргумент - идентификатор на решение което трябва да бъде оценено (или преоценено). Неговата работа е да извлече информацията за това решение, да го компилира, да го изпълни в защитена среда (тъй като кода който се изпълнява би могъл потенциално да съдържа всичко, включително и деструктивни/злонамерени операции) върху входните данни от авторите, да провери резултатите за вярност и да обнови статуса на решението и log-а от изпълнението.

От тези неща се вижда, че този модул е изключително важен за цялата система. За да може в същото време да се реализира бързо, за него се използват голям набор от инструменти налични в Linux.

4.5.7 spoj0-control

spoj0-control е главният инструмент за управление на системата от треньори и администратори. Той представлява cli (Command Line Interface) приложение на Perl изпълняващо различни команди върху системата.

Използването му е подобно на много други инструменти, при които първият аргумент представлява операцията която да бъде изпълнена, а останалите са аргументи към нея. Ето списък на поддържаните команди:

start – стартира оценяващият демон като демон (изисква root права)

start-here – стартира оценяващият демон като на място

stop – спира оценяващият демон

kill – убива оценяващият демон, в случай на зависване

rejudge-problem – преоценява всички неприети решения по дадена задача

rejudge-problem-all – преоценява всички решения по дадена задача

rejudge-run – преоценява конкретно решение

sync-news – синхронизира новините (добавя нови новини записани във файлове)

import-set – импортира тема със задачи

sync-set – синхронизира промените по тема със задачи от файловата система

submit – изпраща дадено решение на дадена задача

help – изпечатва подробно описание на командите

За по-подробно описание вижте 3.5 или извикайте help командата.

4.5.8 spoj0-install

Инсталиращият скрипт е написан на bash и предлага автоматично инсталиране и конфигуриране при Debian GNU/Linux базирани операционни системи. Самият скрипт е последователен (не дава много възможност за избор на настройки) но за сметка на това е прост и дава възможност на потребителите да следят или да се намесват ръчно в работата му, като чака за потвърждение след всяка стъпка.

Скриптът инсталира:

- необходимите пакети от които зависи системата
- системен потребител за работа на системата
- subversion работно копие на системата в home директорията на този потребител, за да се позволява лесно автоматично обновяване, както и изпращане на локално направени промени
- системен потребител за изпълнение на изпратени решения
- базата данни на системата в MySQL и потребител за нея,
- уеб интерфейса като сайт на Apache2 с mod_perl

След инсталация на системата скрипта извършва още:

- стартира тестващият демон
- импортира тестовите теми от системата
- изпраща множество тестови решения от системата

Всички тези стъпки се извършват автоматично. След приключване администраторът може лесно да види дали всичко е наред по статуса на изпратените тестови решения.

Библиография

[1] Wikipedia. Acm international collegiate programming contest — wikipedia, the free encyclopedia, 2007. [Online; accessed 23-April-2007].

[2] Steven S. Skiena and Miguel Revilla. *Programming Challenges.* Springer, May 2003.

[3] Wikipedia. Online judge — wikipedia, the free encyclopedia, 2007. [Online; accessed 23-April-2007].

[4] Urs von Matt. Kassandra: The automatic grading system. Technical Report CS-TR-3275, 1994.

[5] Andy Kurnia, Andrew Lim, and Brenda Cheang. Online judge.

[6] Brenda Cheang, Andy Kurnia, Andrew Lim, and Wee-Chong Oon. On automated grading of programming assignments in an academic institution. *Comput. Educ.*, 41(2):121–131, 2003.

[7] Деветнадесета републиканска студентска олимпиада по програмиране, 2007.

[8] IOI Secretariat. Ioi history, 2007.

[9] Wikipedia. International olympiad in informatics — wikipedia, the free encyclopedia, 2007. [Online; accessed 4-July-2007].

[10] George E. Forsythe. Automatic machine grading programs. In *Proceedings of the 1964 19th ACM national conference*, page 141.401, New York, NY, USA, 1964. ACM Press.

[11] George E. Forsythe and Niklaus Wirth. Automatic grading programs. Technical report, Stanford, CA, USA, 1965.

[12] J. B. Hext and J. W. Winings. An automatic grading scheme for simple programming exercises. *Commun. ACM*, 12(5):272–275, 1969.

[13] PC2 Contributors. An overview of PC2, 2002. [Online; accessed 23-April-2007].

[14] Wikipedia. Pc^2 — wikipedia, the free encyclopedia, 2006. [Online; accessed 23-April-2007].

[15] PC2 Contributors. Obtaining permission to use PC2, 2004. [Online; accessed 23-April-2007].

[16] Wikipedia. Topcoder — wikipedia, the free encyclopedia, 2007. [Online; accessed 23-April-2007].

[17] The SPOJ Team. Sphere online judge (SPOJ), 2007. [Online; accessed 23-April-2007].

[18] Univeridad de Valladolid. online-judge.uva.es - problem set archive. [Online; accessed 04-July-2007].

[19] Univeridad de Valladolid. Statistics for online-judge.uva.es. [Online; accessed 04-July-2007].

[20] Цветан Богданов. Корейската система, 2007.

[21] L. Prechelt, G. Malpohl, and M. Philippsen. Jplag: Finding plagiarisms among a set of programs, 2000.

[22] Kevin W. Bowyer and Lawrence O. Hall. Experience using moss to detect cheating on programming assignments. 2000.

[23] Christian Collberg, Ginger Myles, and Michael Stepp. Cheating cheating detectors.

[24] Cristina Rossi and Andrea Bonaccorsi. Why profit-oriented companies enter the os field?: intrinsic vs. extrinsic incentives. In *5-WOSSE: Proceedings of the fifth workshop on Open source software engineering*, pages 1–5, New York, NY, USA, 2005. ACM Press.

[25] Richard E. Hawkins. The economics of open source software for a competitive firm. *Netnomics*, 6(2):103–117, 2004.

[26] Anthony I. Wasserman and Eugenio Capra. Evaluating software engineering processes in commercial and community open source projects. In *FLOSS '07: Proceedings of the First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*, page 1, Washington, DC, USA, 2007. IEEE Computer Society.

[27] Wikipedia. Debian — wikipedia, the free encyclopedia, 2007. [Online; accessed 23-April-2007].

[28] Ian Murdock. Overview of the debian gnu/linux system. *Linux J.*, 1994(6es):15, 1994.

[29] Wikipedia. Perl — wikipedia, the free encyclopedia, 2007. [Online; accessed 23-April-2007].

[30] Wikipedia. Apache http server — wikipedia, the free encyclopedia, 2007. [Online; accessed 23-April-2007].

[31] netcraft.com. Netcraft web server survey, 2007. [Online; accessed 05-July-2007].

[32] Wikipedia. Mysql — wikipedia, the free encyclopedia, 2007. [Online; accessed 23-April-2007].

[33] Екипът на OpenFMI. Openfmi, 2007. [Online; accessed 05-July-2007].

[34] Sunghun Kim Kai. Webdav based open source collaborative development environment.

[35] GForge. Gforge collaborative development environment cde, 2007. [Online; accessed 05-July-2007].