DOMjudge teamhandleiding

Samenvatting

Hieronder staat de belangrijkste informatie kort samengevat. Dit is bedoeld om snel aan de slag te kunnen. We raden echter ten zeerste aan dat minstens één iemand binnen je team de complete handleiding doorneemt, omdat daarin specifieke details van het jurysysteem staan die ook van belang kunnen zijn op het moment dat niet alles perfect gaat. **WEES GEWAARSCHUWD!**

DOMjudge werkt via een web-interface die je kunt vinden op
`http://example.com/domjudge/team`.

Inlezen en wegschrijven

Oplossingen moeten invoer en uitvoer lezen van 'standard in' (toetsenbord) en wegschrijven naar 'standard out' (beeldscherm). Je hoeft dus nooit een bestand te openen. Zie bijlage A voor een aantal voorbeelden hiervan.

Insturen van oplossingen

Insturen van oplossingen gaat via het command-line programma `submit` dan wel de web-interface:

Command-line
: Gebruik `submit <probleem>.<extensie>`, met `<probleem>` het label van het probleem en `<extensie>` een standaard-extensie van de programmeertaal. Voor de complete documentatie en alle opties, zie `submit --help`.

Web-Interface
: Vanaf je teampagina op `http://example.com/domjudge/team`, selecteer je **submit** en daar kun je een bestand selecteren en insturen. Standaard wordt het probleem uit het deel van de bestandsnaam vóór de punt gehaald en de programmeertaal uit de extensie.

Bekijken van scores, inzendingen, e.d.

Het bekijken van inzendingen, scores en sturen en lezen van "clarification requests" gaat via de web-interface. De knoppen op `http://example.com/domjudge/team` spreken voor zich.

Einde samenvatting.

1 Oplossingen insturen

Het insturen van oplossingen voor problemen kan op twee verschillende manieren: via een command-line interface (het programma `submit`), of via de web-interface. Het kan zijn dat een van beide niet beschikbaar is, afhankelijk van de configuratie van het systeem door de jury. Hieronder worden beide methodes beschreven.

1.1 Command-line: `submit`

Syntax: `submit [opties] bestandsnaam.ext`

Het submitprogramma haalt de naam (label) van het probleem uit `bestandsnaam` en de de programmeertaal uit de extensie `ext`. Dit kan handmatig aangepast worden met de opties `-p probleemnaam` en `-l taalextensie`. Zie `submit --help` voor een compleet overzicht van mogelijke opties en extensies en een aantal voorbeelden. Als deze helptekst niet op één scherm past, gebruik dan `submit --help | more` om alles te lezen.

`submit` zal het bestand controleren en eventueel waarschuwingen geven, zoals wanneer het bestand al lange tijd niet veranderd is of groter is dan de maximale source-code-grootte.

Hierna geeft `submit` een kort overzicht met de details van de inzending en vraagt om bevestiging. Controleer vooral of je het goede bestand, probleem en taal hebt en druk dan op 'y' om de oplossing in te sturen. Als alles goed gaat, zal `submit` een melding geven dat de inzending succesvol is. Indien niet, zal er een foutmelding verschijnen.

Het submitprogramma maakt gebruik van een directory `.domjudge` in de homedirectory van het account. Hier slaat het tijdelijk bestanden op voor inzending en staat ook een logfile `submit.log`. Verwijder deze directory niet en pas hem niet aan, omdat anders het submitprogramma niet meer correct functioneert. Verder kan er een "public ssh-key" van de jury in de SSH-configuratie toegevoegd zijn. Ook deze is nodig voor het functioneren van `submit`.

1.2 Web-interface

Vanaf je teampagina `http://example.com/domjudge/team` kun je oplossingen insturen door naar **submit** te gaan. Daar kan een bestand geselecteerd worden om in te sturen. Verder kan het probleem en de taal van de inzending ingesteld worden. Deze kunnen ook het standaard geselecteerde 'automatisch' blijven; dan wordt geprobeerd het probleem en de taal van de inzending uit respectievelijk de basis en de extensie van de bestandsnaam te halen.

Nadat je op de submitknop geklikt hebt en dit bevestigd hebt, wordt aangegeven of je inzending goed aangekomen is. Daarna staat deze in je lijst met inzendingen.

2 De uitslag bekijken van inzendingen

Op de team-webpagina staat een overzicht van je inzendingen. Dit overzicht bevat alle relevante gegevens: de tijd van inzending, de programmeertaal, het probleem en de status. Hier vind je ook het scorebord met de resultaten van de andere teams.

2.1 Mogelijke uitslagen

Voor een ingestuurde oplossing zijn de volgende uitslagen mogelijk.

CORRECT Je oplossing heeft alle tests weerstaan: je hebt dit probleem opgelost!

COMPILER-ERROR Het compileren van je programma gaf een fout. Bij de details van deze inzending kun je de precieze foutmelding inzien (deze optie kan uitgezet zijn).

TIMELIMIT Je programma draaide langer dan de maximaal toegestane tijd en is afgebroken. Dit kan betekenen dat je programma ergens in een loop blijft hangen, of dat je oplossing niet efficiënt genoeg is.

RUN-ERROR Je programma gaf een fout tijdens het uitvoeren. Dit kan verschillende oorzaken hebben, zoals deling door nul, incorrecte geheugenadressering (segfault, bijvoorbeeld door arrays buiten bereik te indiceren), te veel geheugengebruik, enzovoort. Let ook op dat je programma met een exitcode 0 eindigt!

NO-OUTPUT Je programma gaf geen uitvoer. Let op dat je uitvoer naar standard output schrijft!

WRONG-ANSWER De uitvoer van je programma was niet correct. Het kan zijn dat je oplossing niet correct is, maar let ook goed op dat je de antwoorden precies zoals beschreven uitvoert: de uitvoer moet exact kloppen met de specificatie van de jury!

PRESENTATION-ERROR De uitvoer van je programma verschilde slechts in presentatie (bijvoorbeeld: de hoeveelheid witruimte) met de correcte uitvoer. Dit wordt net als WRONG-ANSWER niet correct gerekend. Deze uitslag is optioneel en mogelijk niet aangezet.

TOO-LATE Helaas, je hebt ingestuurd nadat de wedstrijd al afgelopen was! Je inzending is opgeslagen maar wordt niet verder behandeld.

3 Clarifications

Communicatie met de jury loopt door middel van *clarifications* (verhelderingen), deze komen op je teampagina te staan. Boven aan de pagina de gegeven clarifications, daar onder je *requests* (verzoeken).

Je kan vragen aan de jury stellen door middel van het doen van een “Clarification Request”, de link hiervoor bevindt zich onderaan de clarifications-pagina. Je vraag zal alleen bij de jury aankomen; zij zullen deze zo snel mogelijk en adequaat beantwoorden. Antwoorden die voor iedereen relevant kunnen zijn zullen naar iedereen gestuurd worden.

Als je een clarification ontvangt van de jury, dan wordt dat automatisch gemeld door toevoeging van "(1 new)" in de clarification knop in het menu. Dit wordt automatisch bijgewerkt zonder dat het nodig is de pagina te herladen.

4 Hoe worden opgaven beoordeeld?

Het DOMjudge jurysysteem is volledig geautomatiseerd. Dit betekent dat er (in principe) geen menselijke interactie is tijdens de beoordeling. Het beoordelen gebeurt via de volgende stappen:

4.1 Insturen

Via het `submit` programma of de web-interface (zie sectie 1) kun je een oplossing voor een opgave insturen, zodat hij geüpload wordt naar de jury. Let op dat je de source-code van je programma moet insturen (en dus niet een gecompileerd programma of de uitvoer van je programma)

Dan komt je programma in de wachtrij te staan, om gecompileerd, uitgevoerd en getest te worden op één van de jury-computers.

4.2 Compileren

Je programma wordt op een jury-computer onder Linux gecompileerd. Als je een andere compiler of besturingssysteem gebruikt dan de jury, moet dat in principe geen probleem zijn, maar let wel op dat je geen compiler/systeem-specifieke dingen gebruikt (afhankelijk van de configuratie kun je bij een compileerfout de foutmelding bekijken).

4.3 Testen

Als je programma succesvol gecompileerd is, wordt het gedraaid en de uitvoer vergeleken met de correcte uitvoer van de jury. Er wordt eerst gecontroleerd of je programma correct geëindigd is: als je programma met een fout eindigt en het goede antwoord geeft, krijg je toch een RUN-ERROR! Er zijn een aantal beperkingen die aan je programma opgelegd worden. Als je programma die overschrijdt, wordt het ook afgebroken met een fout, zie sectie 4.4.

Verder moet de uitvoer van jouw programma exact overeenkomen met de uitvoer van de jury. Let dus goed op, dat je de uitvoerspecificatie volgt. In gevallen waarin er niet één unieke uitvoer is (zoals bij floating point-antwoorden) kan de jury een aangepaste beoordeling hiervoor maken.

4.4 Beperkingen

Om misbruik tegen te gaan, het jurysysteem stabiel te houden en iedereen duidelijke, gelijke omstandigheden te geven, zijn er een aantal beperkingen die aan iedere ingestuurde oplossing opgelegd worden:

compile-tijd Je programma mag er maximaal 30 seconden over doen om te compileren. Daarna wordt het compileren afgebroken en levert dit een compileerfout op. Dit zou in de praktijk nooit een probleem mogen opleveren. Mocht dit toch gebeuren bij een normaal programma, laat het dan de jury weten.

sourcegrootte De sourcecode van je programma mag maximaal 256 kilobytes groot zijn, anders wordt je inzending geweigerd.

geheugen Je programma heeft tijdens het draaien maximaal 524288 kilobytes geheugen ter beschikking. Let op dat dit totaal geheugen is (inclusief programmacode, eventuele virtual machine (Java), statisch en dynamisch gedefinieerde variabelen, stack, ...)! Als je programma meer probeert te gebruiken, zal het afgebroken worden, zodat dit een "RUN-ERROR" geeft.

uitvoergrootte Het is niet toegestaan meer dan 4096 kilobytes te schrijven naar standard out of naar standard error. Als je deze limiet overschrijdt, krijg je een RUN-ERROR.

aantal processen Het is niet de bedoeling dat je programma meerdere processen (threads) start. Dit heeft ook geen zin, want je programma heeft precies één processor volledig tot zijn beschikking. Om de stabiliteit van het jurysysteem te bevorderen, kun je maximaal 15 processen tegelijk draaien (inclusief de processen waardoor je programma gestart is).

Mensen die nooit met meerdere processen geprogrammeerd hebben (of niet weten wat dat is), hoeven zich geen zorgen te maken: standaard draait een gecompileerd programma in één proces.

4.5 Java klassenaamgeving

Het compileren van Java broncode wordt gecompliceerd door de klassenaamgeving van Java: er is geen vast startpunt van de code; iedere klasse kan een methode `main` bevatten. Een klasse die `public` gedeclareerd is, moet verder in een bestand met dezelfde naam staan.

In de standaard configuratie detecteert DOMjudge automatisch de hoofdklasse. Anders moet de hoofdklasse "`Main`" heten en een methode "`public static void main(String args[])`" hebben. Zie ook het Java codevoorbeeld in appendix A.

A Codevoorbeelden

Hieronder staan een aantal voorbeelden van code om de invoer van een probleem in te lezen en de uitvoer weg te schrijven.

De code hoort bij de volgende probleembeschrijving: De invoer bestaat uit één regel met daarop het aantal testgevallen. Daarna volgt voor elk testgeval een regel met daarop een naam (één woord). Print voor elke naam de string "Hello <naam>!". Een naam is maximaal 99 karakters lang.

Dit probleem zou de volgende in- en uitvoer kunnen hebben:

Invoer	**Uitvoer**
3 world Jan SantaClaus	Hello world! Hello Jan! Hello SantaClaus!

Let op dat het getal 3 op de eerste regel aangeeft dat er 3 testgevallen volgen.

Een oplossing voor dit probleem in C:

```
1   #include <stdio.h>
2
3   int main()
4   {
5           int i, ntests;
6           char name[100];
7
8           scanf("%d\n", &ntests);
9
10          for(i=0; i<ntests; i++) {
11                  scanf("%s\n", name);
12                  printf("Hello %s!\n", name);
13          }
14
15          return 0;
16  }
```

Let op de `return 0;` aan het einde, zodat we geen RUN-ERROR krijgen!

Een oplossing in C++ kan als volgt:

```
1   using namespace std;
2
3   #include <iostream>
4   #include <string>
5
6   int main()
7   {
8           int ntests;
9           string name;
10
11          cin >> ntests;
12          for(int i = 0; i < ntests; i++) {
13                  cin >> name;
14                  cout << "Hello " << name << "!" << endl;
15          }
16
17          return 0;
18  }
```

Een oplossing in Java:

```
1   import java.io.*;
2
3   class Main
4   {
5           public static BufferedReader in;
6
7           public static void main(String[] args) throws IOException
8           {
9                   in = new BufferedReader(new InputStreamReader(System.in));
10
11                  int nTests = Integer.parseInt(in.readLine());
12
13                  for (int i = 0; i < nTests; i++) {
14                          String name = in.readLine();
15                          System.out.println("Hello "+name+"!");
16                  }
17          }
18  }
```

Een oplossing in Pascal:

```
1   program example(input, output);
2
3   var
4           ntests, test : integer;
5           name         : string[100];
6
7   begin
8           readln(ntests);
9
10          for test := 1 to ntests do
11          begin
12                  readln(name);
13                  writeln('Hello ', name, '!');
14          end;
15  end.
```

En tenslotte een oplossing in Haskell:

```
1   import Prelude
2
3   main :: IO ()
4   main = do input <- getContents
5             putStr.unlines.map (\x -> "Hello " ++ x ++ "!").tail.lines $ input
```