Caia Tutorial for **0.1**
Linux and OSX Distribution

In the Linux and OSX distributions you can find software to help you to develop the CodeCup 2026 game **0.1**. The software is called Caia and it includes three computer players for testing purposes. You can also create your own players and use Caia to coordinate the games between various players.

For the latest information and updates, please see: www.codecup.org

Installation

Copy the tarball to some place in your home directory and start there a shell window. In this window type the following commands:

```
tar -xzf caia_zeropointone_<distro>.tar.gz
caia_install_zeropointone/install_game.sh zeropointone
```

This command will install Caia, as well as all the precompiled components of **0.1** into your home directory. The installation installs the following files:

- In the folder *caia/zeropointone/bin/*: *manager.txt, caiaio, manager, competition, competition.sh, referee, player1, player2, player3, javawrapper, jarwrapper* and some content in the folder *caia/zeropointone/bin/resources/*.
- In the folder *caia/zeropointone/src/manager/: manager.cc, Makefile*.
- In the folder *caia/zeropointone/src/competition/: competition.cc, Makefile*.
- And three empty folders: *caia/zeropointone/refereelogs, caia/zeropointone/playerlogs* and *caia/zeropointone/competitionlogs/*.

The software is now ready to use.

Playing a game

To play games between player1 and player2, just execute from the command line in the *caia/zeropointone/bin/* folder:

```
./caiaio
```

Once the game is finished you can find log files of the referee and players in the directories *caia/zeropointone/refereelogs/* and *caia/zeropointone/playerlogs/*. The referee produces an html file so that you can watch the game. Note that the playerlog file only contains information written by the player to its stderr. (The two example players only write an ID to their stderr).

It is also possible to see exactly what is happening between the manager, caiaio, the referee and the client players. To see this, start caiaio in debug mode:

```
./caiaio -d
```

This option is useful if you let your own player print evaluation or runtime data about each move to its stderr so that you can debug more easily your player in case of failure.

The referee produces an html file that has a link to the javascript files in *../bin/resources/*. You can change this default link with another path. The only thing you have to do is to make a *referee.txt* in which you refer to the javascript files with the new path:

```
jspath=../newpath/
```

The manager

The source code of the manager and the makefile are included in the distribution. You can change *manager.cc* according to your own wishes. For more information we refer you to the Caia documentation.

The information on what games will be played is stored in *caia/zeropointone/bin/manager.txt*. To play two games, player1 against player2 and player2 against player3, you can change it to:

```
2
player1 player1first.log player2 player2second.log
player2 player2first.log player3 player3second.log
```

The competition manager

Now a competition manager is written for organizing your own competitions in a very simple way. The software is part of Caia and is using the caiaio in a similar way as the manager does in the distros. Now it is possible to test your own players easily and find out which one of them is strongest.

For the latest information and updates, please see: www.codecup.org

Running a competition

To run a competition between several players just copy the players in the *bin/* folder. Make sure that there are no other files in the folder than necessary to run Caia. Each unknown file will be regarded as a client player! Only the files with the names *javascript*, *caiaio*, *manager*, *referee*, *competition*, *generator*, *javawrapper, jarwrapper, *.txt, *.class* and **.jar* will be discarded as such. Then execute from the command line in the *bin/* folder:

```
./caiaio -m competition
```

Once the competition is finished you can find log files of the referee, the players and the competition in the directories *caia/zeropointone/refereelogs/*, *caia/zeropointone/playerlogs/* and *caia/zeropointone/competitionlogs/*. The referee produces an html file so that you can watch the game. Note that the playerlog file only contains information written by the player to its stderr. The information about the played games for future competitions is stored in the *competitionlogs/* folder.

It is also possible to generate a log file with all game results in the *competitionlogs/* folder. The place and the name of the log file can be chosen freely. Example:

```
./caiaio -m competition -f ../competitionlogs/logfile.txt
```

Running a new competition with a new player

All you have to do is to add a new player to the *bin/* folder and rerun the competition. Games that were played earlier will not be replayed. Mind that each time an existing player is modified, the games of this player will be replayed!

This is how our competition manager works

In general you can have different type of players: players that always do the same and the so-called random players. To get a good estimate of the strength of such a random player it is better to play 50 games and then to output the mean score by calculating the sum of the 50 scores divided by 50.

If you wish to let the competition manager work in another way, just change de source code in *caia/zeropointone/src/competition/competition.cc* and recompile.

Running competitions using competition.txt

For those of you who prefer to have more control over their competitions can make use of a text file. All you have to do is to put a text file *competiton.txt* in the *bin/* folder with similar content:

```
3
david2
marcel11
ludo3
```

In the example a full competition will be run among the client players *david2*, *marcel11* and *ludo3*. Other players in the *bin/* folder will not participate.

Running competitions using competition.sh

Since the CodeCup competition 2015 an alternative way is provided to run competitions. The advantage of this alternative is that it makes use of all the cores in your pc. All you have to do is to run the script *competiton.sh* in the *bin/* folder:

```
./competition.sh david2 marcel11 ludo3
```

In this example a full competition will be run among the client players *david2*, *marcel11* and *ludo3*. Other players in the *bin/* folder will not participate. Under Linux and OSX the number of cores is determined automatically.

You can adjust the number of processes manually. This can be done from the command line:

```
num_cores=3 ./competition.sh david2 marcel11 ludo3
```

In this example 3 processes will run parallel. Default is the number of cores and should not exceed the actual number of cores available in your pc unless you want to test with it. By making use of hyper threading competitions run slightly faster (up to 30 %), but will introduce time measurements mistakes made by the caiaio. Two folders *competition-results/* and *competition-stubs/* will be added. The results of the games are stored in *competition-results/* so that in new competitions games between existing players are not replayed again. The folder *competition-stubs/* contain the batch files used for the multicore scheduling. Like with the conventional competition manager the player logs and referee logs are stored in *playerlogs/* and *refereelogs/*.

If you move (or have moved) your files to another folder, you must edit line 8 in the script *competiton.sh* in the *bin/* folder:

```
BASE=$HOME/caia/zeropointone
```

Mind that the caiaio in this case is measuring the real time execution of your program. So you should compare the execution time of your program on the CodeCup system with that of your system. The time limit (30000 milliseconds) can be changed in line 15 of the script:

```
PLAYER_TIME_LIMIT=30000
```

Working with Java classes and JAR files

In the distribution of the tarballs is the source code included for the javawrapper and the jarwrapper. These two programs let your Java player work with Caia.

If you have made a Java program *JavaPlayer.class*, you must do two things. The first thing you'll have to do is to put this file into the *bin/* folder. The second thing you'll have to do is to rename the program *javawrapper* in the *bin/* folder into *JavaPlayer*. Then it should work. More information is found in the header of *javawrapper.c*.

If you have made a JAR file *JavaPlayer.jar*, you must do two things. The first thing you'll have to do is to put this file into the *bin/* folder. The second thing you'll have to do is to rename the program *jarwrapper* in the *bin/* folder into *JavaPlayer*. Then it should work. More information is found in the header of *jarwrapper.c*.