

Giocoso

The Randomising FLAC Player



by Howard Rogers

Version 1.1 – November 2021

Giocosso User Manual

by Howard Rogers

Copyright © 2021 Howard Rogers

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the express prior permission of Howard Rogers (hjr@absolutelybaching.com)

The Giocosso program, together with this accompanying documentation, is distributed in the hope that it will be useful, but **without any warranty**; without even the implied warranty of **merchantability** or **fitness for a particular purpose**.

In no event shall liability for any direct, indirect, punitive, incidental, special consequential damages, to property or life, whatsoever arise out of or connected with the use or misuse of Giocosso or this accompanying documentation.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries. All other trademarks are the property of their respective owners. Howard Rogers is not commercially or financially associated with any product or vendor mentioned in this publication.

Table of Contents

1.0	Introducing Giocosos	6
1.1	Some Prior History	6
1.2	The Name	7
1.3	What can Giocosos play?	7
1.4	Why Giocosos?	8
2.0	Prerequisites for running Giocosos	11
2.1	Giocosos's Digital Storage Expectations	11
2.1.1	The Folder is Key	12
2.1.2	File Numbering/Naming affects play order	12
2.1.3	Tag Fields and Appropriate Data	14
2.1.4	Giocosos and the Axioms of Classical Tagging	15
2.1.5	Music Ownership	15
3.0	Installing Giocosos	16
4.0	Giocosos – First Run	17
5.0	Giocosos's Play Modes	19
5.1	Direct Play Mode	19
5.2	Random Play Mode	20
5.3	Summary of Play Modes	22
5.4	Pausing and Terminating Play	23
6.0	The Giocosos Program Display	24
6.1	Play Window	24
6.2	Album Art Display	25
7.0	Integrating Giocosos with the Desktop	28
7.1	KDE - Plasma	28
7.2	XFCE - Thunar	29
7.3	Gnome - Nautilus	29
7.4	Budgie	30
7.5	Other Desktop Environments	30
7.6	Remembering Album Art Display Location	30
8.0	Scrobbling	32
8.1	Configuring Scrobbling	32
8.2	Getting Giocosos to Scrobble	35
8.3	How Giocosos Scrobbles	37

8.4 Scrobbling with Embedded Cuesheets.....	37
9.0 The Giocosso Database.....	40
9.1 Creating a Database.....	40
9.2 Refreshing a Database.....	41
9.3 Table Structures and Methodology.....	42
9.4 Schema Description.....	43
9.5 Accessing the Database.....	45
9.6 Backing up the Database.....	45
9.7 Recovering the PLAYS table.....	46
10.0 Run-Time Parameters.....	47
10.1 Initialization Parameters.....	48
10.1.1 Createdb.....	49
10.1.2. Refreshdb.....	49
10.1.3 Musicdir.....	50
10.1.4 Importplays.....	51
10.1.5 Scrobble-config.....	51
10.1.6 Checkver.....	52
10.1.7 License/Licence.....	52
10.2 Appearance/Behaviour Parameters.....	53
10.2.1 Displaycolour/Displaycolor.....	54
10.2.2 Artisize.....	56
10.2.3 Captiontext.....	57
10.2.4 Captionbackground.....	57
10.2.5 Listcolours.....	58
10.2.6 Scrobble.....	58
10.2.7 Device.....	59
10.2.7 Editor.....	60
10.3 Selective Parameters.....	61
10.3.1 Dbname.....	63
10.3.2 Selections.....	64
10.3.3 Pause.....	64
10.3.4 Timebar.....	64
10.3.5 Genre.....	65
10.3.6 Composer.....	66
10.3.7 Performer.....	67

10.3.8 Comment.....	67
10.3.9 Composition.....	67
10.3.10 Recordnumber.....	68
10.3.11 Unplayed.....	69
10.3.12 Unplayedworks.....	69
10.3.13 Minduration.....	70
10.3.14 Maxduration.....	70
10.3.15 Negate.....	70
10.4 Reporting Parameters.....	72
10.4.1. Report.....	73
10.4.3 Reportsort.....	75
10.4.2 Reportdays.....	76
10.4.4 Reporttype.....	76
10.4.5 Recordinglist.....	77
10.4.6 Composername.....	78
10.4.7 Stats.....	79
10.5 Run-time Parameter Summary.....	80
11.0 Some Technical Extras.....	81
11.1 Use of ALSA.....	81
11.2 Lost Scrobbles.....	81
11.3 Fonts.....	84
11.4 Aliases.....	84
11.5 Transitioning from AMP to Giocosos.....	87
11.6 Tested Distros.....	88
Appendix A: Differences between Giocosos and AMP.....	91
Appendix B: The Requirement for Sudo Privileges.....	94
Appendix C: Product Registration.....	96
Appendix D: Giocosos on Windows.....	97
Acknowledgements.....	98

1.0 Introducing Giocososo

Giocososo is a gapless, trackless, optionally-randomising, optionally-scrobbling command-line FLAC player for Linux.

It is minimalist in design and in resource consumption, aiming always to get out of the way and let no distractions get between you and playing your classical music collections perfectly.

It runs in two modes: either with a database, in which case it will randomly select things to play that meet various selection criteria you can send its way; or directly from a folder full of FLACs, in which case whatever music is present in that folder will be played, continuously, from beginning to end.

In randomised database mode, Giocososo can be directed with various command-line switches to ‘tweak’ or adjust its randomising behaviour, so that music of particular composers or of particular genres can be played; or music that lasts a certain length of time can be selected. If music files have been tagged with embedded album art, Giocososo will display this album art at various sizes, unless instructed not to do so.

Giocososo can be instructed to submit details of what it plays to [Last.fm](https://last.fm), a process known without especially good reason(!) as “Scrobbling”. This allows you to build up a web-accessible history of your music listening over time.

If run with a back-end database, Giocososo will store details of what it plays in its own database, in a PLAYS table, and the contents of that table can be queried by running Giocososo in one of its various reporting modes. These reports can be displayed on-screen or written to disk in pipe-delimited text files. Such files can be imported into spreadsheets of other database back-ends quite easily, allowing for in-depth analysis of one’s music collection and listening habits over time. See my own website at <https://absolutelybaching.com/music-plays-per-composer/> for an example of what sort of thing is possible in this regard).

1.1 Some Prior History

Giocososo is not the first minimalist, command-line etc etc FLAC player I’ve written: I prototyped a previous one called ‘the Absolutely Baching Music Player’ or AMP for short at the very end of 2020. After six months of bolting on new bits of functionality as I came across a need for them, I decided that AMP had reached the end of its useful life in June 2021. Giocososo draws heavily on AMP’s code base and could indeed be thought of as ‘AMP v.2’... but it also makes several fundamental changes to AMP’s way of working, so it’s a bit more than a version 2 revamp.

If you have previously been using AMP to play your music collection, you will find some run-time parameters have disappeared; the default is now not to scrobble, so that needs to be enabled every time Giocososo runs; and some new reporting options have been devised. I detail the differences between AMP and Giocososo more thoroughly in **Appendix A**.

Giocososo is forwards-compatible with AMP however -meaning, specifically, that if you have an extensive history of music ‘plays’ recorded in AMP, you can bring that forward into Giocososo and not lose it. For instructions on doing that, see **Section 9.7** on ‘Recovering the PLAYS table’: the steps

explained there allow you to restore a Giocosos play history from backup, but also apply equally well to ‘restoring’ an AMP play history into a new Giocosos database.

1.2 The Name

As a classical music fan, you will probably already know that the word *giocosos* is a tempo indication that means ‘lively, humorous, joyful’ or, particularly, ‘**playful**’. I couldn’t think of a better name for something designed to ‘play’ my music collection.



On a slightly less cerebral note, a random search for ‘giocosos’ one day brought forth pictures of a party-mode cat. Adoring cats almost as much as I adore music (and wine!), I thought the pairing was ideal.

Joe Playful (and his feline brethren) are thus the official mascots of Giocosos. They will be used relentlessly in all future documentation and news releases about the program!

Meanwhile, regardless of why the name came about, draw inspiration from the party-loving cats and feel free to use Giocosos to listen to your classical music collection in new, insightful and, above all, playful ways!

1.3 What can Giocosos play?

Giocosos can only play FLAC files. Presented with MP3s, WAVs, OGGs, APEs, AC3s or any other music format at all, it will simply ignore them: it is hard-coded to look for things called ‘*.flac’ and anything that doesn’t have a .flac extension will simply be ignored. It doesn’t even matter if a file is a genuine FLAC file internally: if the file extension doesn’t say ‘.flac’, it will be invisible to Giocosos.

Giocosos does not, of course, know whether it is playing a FLAC of *classical* or *popular* music: it makes neither moral nor artistic judgments in that regard! So, what types of FLAC files you present it with makes no real difference to its functionality.

That said, the *way* it plays music files is probably more appropriate to classical music than to other sorts of music. I mean, simply, that it plays folders of music at a time, without pausing between tracks, and without providing any possibility of the user pausing the play either. Now, it may be perfectly standard practice to play Pink Floyd’s *Dark Side of the Moon* as a whole album in one sitting, without pause... in which case, Giocosos will fit in fine. But it is certainly standard practice in classical music circles to play all four movements of a Beethoven symphony one after the other, without significant pause between them, and without interruption by the audience... so Giocosos’s mode of operation seems best suited to that sort of ‘concert hall’ music listening experience, as far as I am accustomed to it.

Ideally, you will present a nice set of tagged FLAC files to Giocososo: it will then be able to display what it's playing as it plays it, because it can read those tags and use the information they contain to alter its display. If your music is not well-tagged, however, then it will still be able to play it directly, but the display will be annoyingly devoid of useful information!

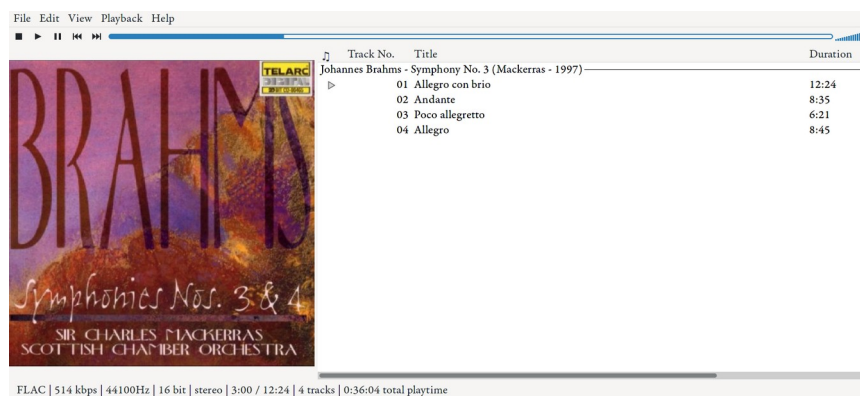
More seriously, if your music is not properly tagged, it cannot meaningfully be scanned and read into a database -and, at that point, Giocososo cannot use its database back-end to randomise the playback of those music files. Without decent tagging, therefore, Giocososo's functionality is seriously restricted: I mention more about this in some detail in **Section 2.1** below.

1.4 Why Giocososo?

Linux is not short of media players of all sorts and capabilities! From the ubiquitous **vlc** to the oddly-named but excellent **DeaDBeeF**, there is a media player to cater for, apparently, everyone's whims and wishes. Except for mine, it would seem, though: I do use other media players from time to time, but none of them were entirely "right" as my main player of classical music. I think perhaps it might be useful if I lay out the main reasons why I thought it necessary for me to write a new music player, rather than use someone else's!

In no particular order, I'd list my motivations as follows:

- Every music player under the Sun displays -and plays- "tracks". Even really good ones (such as DeaDBeeF, shown here) have this sort of display:



Now it's nice to see that there are four movements of Brahms's 3rd Symphony, I suppose... but what actual *use* is there in that display? Does a listener of classical music ever think, 'I know Brahms wrote a 4-movement symphony, but I only want to listen to the third movement?' I mean, I guess they *might* do... but *should* they? In my view, they shouldn't. Composers wrote whole operas or whole symphonies or whole concertos: we should listen to them as entire pieces, not just pick the bits from them we like and play in any order. Thus, I wanted my music player to be 'composition-aware' and 'track-oblivious'. If you like, I wanted a player that would create a music listening experience more akin to going to a concert (and having to sit there from beginning to end, whether you like it or not!) than listening to a digital music player with 'tracks', 'playlists' and 'top 30s'.

- Over the years, I have got into the habit of over-playing some composers and under-playing others, creating a seriously unbalanced listening experience. You can certainly justify it: my top composers were my favourites because they wrote the ‘best’ music, after all. But it must equally be plausible that I was barely giving hundreds of composers even a *chance* to impress. So I wanted a music player that chose music *for me*: that would pick things at random and present them to me to listen to, whether I had them in mind or not. A music player where you have to scroll through long lists of composers or compositions and actively pick something to play would not fit this bill. I needed a *randomising* player -though one where I could override the random element if I wanted to, as well.
- I needed a player that would (a) scrobble (that is, add to the record of ‘plays of music’ I’ve made to my account at Last.fm) and (b) would maintain its own record of what I’d played, in a database, from which I could extract data to analyze and explain patterns in my music listening history. A lot of Linux music players will do (a), but they don’t often do (b) -or, if they do, they don’t advertise it as a key feature that’s readily accessible.
- I wanted a player that spoke the language of classical music: that labelled things as ‘composers’, ‘compositions’ and ‘movements’, not ‘artists’, ‘albums’ and ‘tracks’. Now, a lot of music players will make room for a column called ‘composer’: but they won’t scrobble it as such, nor allow it as the basis of a selection. I wanted composers and compositions to be front and centre of the music listening experience.
- No classical music player should add anything to the digital audio stream, or interfere with it in any way: we care about our music too much for such tricks to be desirable! So players that include ‘graphics equalizers’ or ‘sound effects’ to make it sound as if you’re listening to a recording in a concert hall are not good things! Instead, I want a player to pipe its output directly to a hardware device without intervening in its processing. That also rules out a volume control -and pause/play buttons. If the player can control those aspects of the music stream, they’re interfering too much! I have a proper hardware amplifier with a volume control knob to use for such things, after all ☺
- Classical music has a particular fondness of the *attacca*, whereby one movement starts abruptly as the previous one finishes: any decent classical music player needs to be able to play that transition between movements *without introducing an audible gap*. In fact, proper **gapless playback** is an essential quality of a decent music player, whatever type of music its playing, but it’s especially important to classical music listeners. Other players can do this to various degrees: many even do it perfectly. Mine had to be at least as good to compete (and it is!)
- I’ve spent a lot of time physically organising my music collection in a *composer* → *genre* → *composition* folder hierarchy. I have equally spent countless hours carefully tagging my music files with appropriate tag metadata, so that I can distinguish my Leopold Mozart from my Wolfgang, my Max Bruch from my Max Reger... and so on. My music player needs to respond to this logical and physical organisation, and use it constructively to make clear music-playing decisions. If I want to listen to ‘ballet’, or ‘anything sung by Callas’ or ‘something not conducted by Karajan’, I should be able to instruct my player accordingly and have it work out, dynamically, what music files then count as ‘valid playback candidates’.

- I am usually listening to music whilst doing other things -which is definitely *not* to say, my music becomes ‘background muzak’: those ‘other things’ might well be me trying to follow the music in score, for example. Anyway, no matter the specifics, it means I want my music player to get out of my way: to play the music and then vanish, instead of using ‘look at me!’ visual tricks. For my money, that means ‘command line’ and ‘no fancy graphics’. So, all those music players that offer ‘visualizations’ to entertain they eyes as the ear is distracted away from the important stuff... they’re out! Now, you can always minimise a GUI player and keep it tucked away... but nothing gets more minimal than running a command-line tool in a ‘dropdown’ terminal, such as are provided by the likes of Guake or Yakuake. So, terminal, command-line, text-based it had to be. Yes, my PC has a 10-core, 20-thread CPU with 96GB of RAM and 16TB of mirrored hard disk space, along with a 44” 4K monitor: I’m going minimal because that’s what classical music deserves, not because I’m hardware-constrained!
- Classical music demands you pay it serious attention, as I’ve already mentioned. For my money, that means it needs to be played in the highest possible quality... and that means, MP3s and OGGs and similar lossy audio formats are not acceptable. My music player therefore needs the ability to plays FLACs (including high resolution FLACs, equivalent to what you’d get from an SACD) and nothing else. Players that can handle a bazillion crappy formats I’m never going to want to subject myself to... they’re out!
- To a large extent, I have “CAO’d” my classical music. In other words, I have used my own Composition-At-Once program to turn a multi-file composition into a single-file with an embedded cuesheet that describes where and when all the individual movements can be found ‘within’ that single file. I might rip a symphony CD into four separate tracks, tag each of them up with their individual movement details, and the run CAO against them and convert them back into a single, large FLAC file, with an embedded cue sheet that describes precisely where each movement starts and stops and what its tempo markings are. My music player needs to be able to read such embedded cuesheets and thus capture details of the individual ‘tracks’ I’ve played, even though there’s physically only one file to play. Giocosos does exactly that. If it’s playing a folder containing four individual tracks, it will declare that it’s playing ‘4 tracks’. If it’s playing a folder containing one ‘aggregate’ file that contains an embedded cuesheet, it will say it’s playing ‘4* Tracks’ ...and the asterisk tells you that they’re not “real” tracks, in the sense of being physical files on disk. This ability to switch between per-file and per-composition ways of thinking is vital in respecting the way I approach, think of and work with my music. It is a pretty uniquely-classical way of thinking of music, I think.
- I like to analyze how I interact with music: if I’ve got 10,000 recordings I’ve not yet listened to, I need to know that fact, so I can do something about it, for example. That means I need to be able to know what I’ve played, when, how long it lasted and so on. I need my player to be able to store play data which tells me these sorts of things when easily interrogated. I don’t know *any* players on Windows or Linux that can do that very well: but Giocosos does.

Now, if *any* of the above paragraphs ring a bell with you, then you’re the target market for Giocosos. Anyone that cares about classical music, wants to know what they’re listening to in the aggregate, and want to listen to music properly organised and in the highest audio quality: Giocosos is for you!

2.0 Prerequisites for running Giocososo



There are some obvious things you need to do before you can successfully run Giocososo to play your music. You must, for example, be using Linux. You also need to be able to run `sudo` commands when needed (describing how to achieve that on various distros is discussed in **Appendix B** of this document).

You must also store your digital music files in the lossless FLAC format: Giocososo only plays FLAC and will not (by design!) play APE, WAV, MP3, OGG or any other digital music format. It was designed to play classical music: it takes the view that classical music needs to be listened to in as high quality as possible. Lossless formats are the only way to do that. It's also a piece of open source

software and therefore was designed to play open source audio formats only: the Free, Lossless Audio Codec seemed to invite itself to the party, given the huge clues in its name!

There are some more specific software prerequisites too: you'll be told about them when you first try to run Giocososo on a system where they have not been installed previously. Probably the most important component is **ffmpeg**, which is available for all Linux distros (but may require some finagling to install on some distros, such as openSUSE and Fedora, which seem to think it might be patent encumbered and thus don't include it by default in their 'standard' repositories). In any case, as I say, the program itself will warn you about necessary software dependencies when it is first run.

By far the most important prerequisite for running Giocososo successfully, however, is so obvious it perhaps goes without saying, but I'll say it anyway: you need a well-organised digital music collection, appropriately and thoughtfully physically structured, with appropriate metadata tags.

It's actually this last prerequisite which will trip most people up: it seems that rather a lot of people who are otherwise serious about listening to their classical music cannot be bothered to organise or tag their music properly. I've seen examples where the files are just stored in a single folder, for example, with all key data (such as composer, composition name and the artists making the recording) are somehow 'encoded' into the physical file name.

Giocososo *will* play such music collections, because it will see the digital files and happily process them. But the results will not be the flexible, controllable, useful program you probably hoped for.

2.1 Giocososo's Digital Storage Expectations

So let me be explicit about what sort of music storage Giocososo *will* work brilliantly with! I'd summarise it in three main points, I think:

- Files are organised in a **Composer > Genre > Composition** physical folder hierarchy
- Files are tagged with a track number, and the track number becomes part of the physical file's name, so that the files in a folder are physically organised in the order in which they should be played

- Files are tagged in such a manner that **Artist=Composer**, **Comment=Performers**, freely described, and **Performer=the main, principal and distinguishing performing artist**, described simply as a single name.

2.1.1 The Folder is Key

I think the first point here is the fundamental and crucial one. Giocosos plays *folders*, not files, so each folder in your collection needs to represent a *single, unique* ‘unit of play’, which is usually a composition. By this, I mean that whilst you might buy a CD containing both Beethoven’s 4th and 5th symphonies, if you rip it as a single ‘thing’ into a single folder, then Giocosos will subsequently play eight movements from beginning to end, without significant pause between any of them. You’ll have just invented Beethoven’s 4to5th Symphony as a single ‘composition’... which is conceptually just wrong!

What you actually need to do instead, if you are aiming at musical sense, is to rip half of that CD into one folder and half into a separate folder. Two folders, each containing the music of a single Beethoven composition. Now Giocosos can play either Symphony No. 4 or Symphony No. 5... but it won’t end up playing both of them in one sitting, just because you didn’t rip things sensibly...

Putting it even more broadly, then: you need to stop regarding the CD as anything significant in your music collection. It’s not important in the slightest: it is merely the packaging in which the stuff that actually counts -*compositions*- are supplied to you. Your job on receiving a new CD is to rip it in a way that sensibly separates out separate compositions into their own, unique storage folders.

Sometimes, though, instead of separating things out into separate folders, you’ll need to do the reverse: join separate things together. Wagner’s *Götterdämmerung* is usually supplied on 4 separate CDs, for example. If you rip them into 4 separate folders, Giocosos will end up playing four separate things in no particular order -and I think it would be a shame for Brunhilde to immolate herself before the Norns have even opened proceedings.

In cases such as long operas and oratorios, then, you may well start by ripping each CD to a separate folder, but you will need to combine them into a *single* folder to end up with a single physical ‘thing’ which Giocosos can play as a single composition.

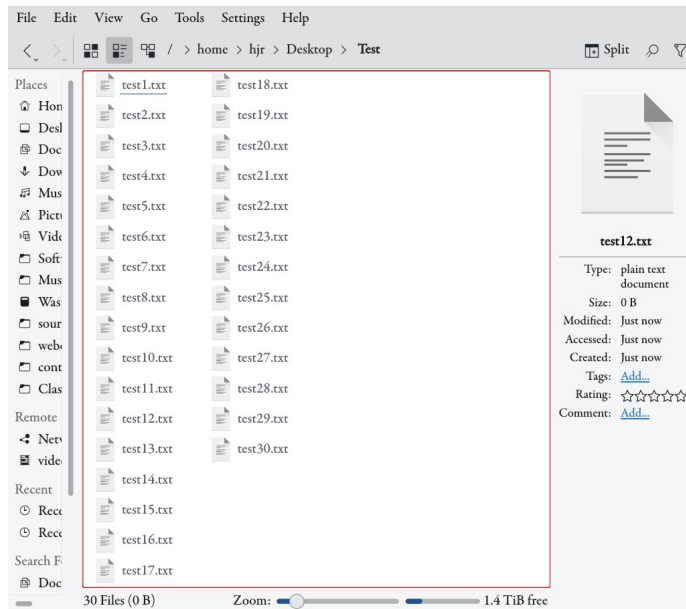
However you look at it, it’s the *physical* folder structure that Giocosos will use to define, navigate and understand your music collection. It will use the *logical* metadata tags you also supply to record what it *has* played and to display useful information about what it *is* playing currently... but it’s the *physical* folder structure that separates your music out into ‘units of play’ that Giocosos will understand and ‘see’.

2.1.2 File Numbering/Naming affects play order

My second bullet point above is important for determining the order in which Giocosos plays things *within* a physical folder. It does it, essentially, by alphabetical sorting of file names -and, in alphabetical sorting, 1 comes before 2, which comes before 3 and so on. So, provided your physical track names are things like “1 – Zitto!”, “2 – Ahimé!” and “3 – Grand Dio!”, then the music will play in 1-2-3 order, no matter that the ‘word part’ of their names is listed in Z – A – G order. Now, 1, 2 and 3 will *also* be the TRACKNUMBER tag for each file: that’s the FLAC metadata tags kicking in, and it’s good that you also logically tag your music files into the correct order too. But: fundamentally, Giocosos is paying

attention to the *physical file names* when it's ordering things, not their metadata track number tags. There should definitely be a *correspondence* between the logical and the physical... but it's the physical that is king as far as Giocosio is concerned.

Be warned, however, that where we humans see “1,2,3,4,5...10,11,12,13...19,20,21...” as ascending numerically, computers generally do not -though you may not even notice this fact when looking at things in your file manager:



That looks sorted perfectly normally, even if it's been split over two columns, right? Well: that's just *one* program managing to do it correctly... but take a look at what my **file system** thinks of those names:

```
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find

[hjr@britten Test]$ ls -l
total 0
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test10.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test11.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test12.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test13.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test14.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test15.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test16.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test17.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test18.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test19.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test1.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test20.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test21.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test22.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test23.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test24.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test25.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test26.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test27.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test28.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test29.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test2.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test30.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test3.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test4.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test5.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test6.txt
-rw-r--r-- 1 hjr hjr 0 Jun 22 11:12 test7.txt
```

As you'd expect, things that are numbered 1 get sorted before things numbered 2 and 3. But imagine the file names being listed out as if you were doing basic arithmetic with them. You'd probably right-align them and thus write them out as:

```
1.txt
10.txt
```

...and at that point, the leading '1' of the '10.txt' item is being compared to 'nothing at all' at the front of the '1.txt' item: and 'nothing' is always sorted *after* numbers by a file system. From the previous screenshot, therefore, you see the consequences: the inconsistent lengths of the numbers in the file names therefore causes 1.txt to sort *after* 10.txt. Similarly, '2.txt' gets sorted after all the files that have names numbered in the twenties.

The file system therefore lists (and Giocosio will therefore play) tracks 10, 11, 12, 13...19 first; then track 1; then tracks 20, 21, 22...29; then track 2; then tracks 30, 31... and so on.

Putting it succinctly: inconsistent track number lengths will muck up your playback order.

In a three-movement concerto, then, you'll get away with track numbers '1,2,3'. In a four-movement symphony, you'll also get away with track numbers '1,2,3,4'. But if you're looking at a Wind Serenade with 12 movements, you'll get into trouble with track numbers '1,2,3,4...' and should instead be naming and tagging them as tracks '01, 02, 03, ... 11, 12': everything consistently 2 digits long.

For the sake of not having to remember to do things different for some works as compared to others, though, I'd basically suggest you get into the habit of *always* using track numbers that are consistently two digits long, padded with leading zeroes if needed (and be prepared to make an exception for the very rare composition that has more than 99 'tracks').

As a final tip: *always* check your file ordering at the file system level, not in a file manager: if it's wrong in a terminal, it will be played in the wrong order by Giocosio, too.

2.1.3 Tag Fields and Appropriate Data

As to the third of my earlier bullet points, where I suggest you use the ARTIST tag to store the music's composer, and the COMMENT tag to store its performers.... Well, you don't have to agree with those suggestions, but they do represent the way Giocosio works! So if you decide to tag things your own way, you'll find a lot of Giocosio functionality won't work for you properly. For example, as you'll go on to discover, you can ask Giocosio to "play me some Britten" by running it with a `--composer=britten` parameter. But that only works if the name 'Benjamin Britten' has been stored in the ARTIST tag. Store your composer names somewhere else, and that parameter will not work properly.

In other words, Giocosio is written expecting to find certain classes of information in certain FLAC tags: if you have chosen to put different information in different tags, Giocosio can still be used to *play* your music, but you won't be able to exploit its searching and filtering capabilities in the way they were intended.

2.1.4 Giocososo and the Axioms of Classical Tagging

If in doubt about any of these ‘music organisation principles’, you should read my article on the ‘Axioms of Classical Music Tagging’, available at <https://absolutelybaching.com/music-articles/the-axioms-of-classical-music-tagging/>. That outlines and explains in detail how you should tag your classical music files -and why. If you at least follow Axioms 1 to 10, your music collection will be well-suited to being played by Giocososo.

By that I mean that Axioms 11 upwards are more ‘matters of taste’ than core organisational principles. Thus, personally, I like to catalogue my Beethoven under ‘L’ and my Mahler under ‘G’, as per the dictates of Axiom 12... but if you think that’s an insane way of doing things, you are free to ignore that particular way of cataloguing music! (But keep in mind before you dismiss Axiom 12 that Giocososo will find Mahler wherever you file him, provided only that the ARTIST tag mentions him somehow: following Axiom 12 doesn’t make your music any hard *for a computer* to find, basically).

Remember, too, that if the demands of Axioms 1 to 10 seem oppressive, and far too much work to get right, you can always cheat: Giocososo does not care about track titles, so if you call everything you own ‘track01, track02’ and the like, it will play (and search for) things just fine. I can nevertheless not help feeling that you owe your collection better care and curatorial management of it than that, so I wouldn’t personally recommend that approach!

2.1.5 Music Ownership

It seems appropriate now to mention a final core assumption that Giocososo makes about your music collection: that you *own* it and that it is stored as files on a hard disk in your physical possession! In the second decade of the twenty-first century, I realise, ownership of music can seem a quaint, archaic thing. We have the likes of Spotify and Qobuz telling us that one can listen to music just as effectively and pleasurably by ‘renting’ *their* copies of music as one can by going to all the faff and hard work of owning your own music collection. Why not simply rent your music, then?

Well: the first thing to say by way of rebuttal to that proposition is that if you do, Giocososo is not the music player for you. It has zero ‘cloud playing’ capabilities, so anything you ‘own’ in the cloud is inaccessible to it, let alone anything you merely rent from the cloud.

And the second thing I’d add is that I can go to the Post Office and buy as many stamps as I like -but that doesn’t make me a stamp collector. I can similarly go to the supermarket and get as many small coins in my change as I can handle, but that makes me no coin collector! If you treat your classical music as a commodity to be consumed, a metaphorical pile of loose change, be my guest... but Giocososo is not the music player for you. There’s nothing wrong with buying stamps to post letters, or using small change to make small shop purchases -but neither task makes you a philatelist or a numismatist!

Likewise, Giocososo is intended for use by people who regard classical music as something to own, collect and carefully curate... and not merely to consume.

3.0 Installing Giocososo

To install Giocososo on your Linux system, simply type these commands at a terminal prompt:

```
cd
wget https://absolutelybaching.com/abc_installer
bash abc_installer --giocososo
```

The **cd** makes sure what is about to happen takes place in your own home folder. The **wget** command fetches the Absolutely Baching Installer script (and obviously needs to have been installed before you start: it's present by default on most Linux distros with which I'm familiar, though, so should generally work 'out of the box'). The **bash** command then runs that script.

The installer script will fetch the Giocososo software from the Absolutely Baching website and will copy it into the appropriate folders for you. Since root permissions are needed to do this copying, the script will prompt you to supply your sudo password at one point: you do need to be able to run with root permissions via sudo for the installation to succeed.

An optional step is now to **make sure that your terminal of choice always opens with dimensions of at least 102x26**. Most distros seem to default their terminals to opening at 80x24, and this will result in garbled output from Giocososo. So you should alter the 'profile' governing a terminal's default size. On Konsole, that's *Settings* → *Edit Current Profile*, but you also need to do *Settings* → *Configure Konsole* and turn **off** the 'Remember window size' option, so that the new profile values take effect. In Gnome, it's just click the terminal's 'hamburger menu' (three horizontal lines, up in the top left), then click *Preferences*. The default profile is named under the 'Profiles' item on the left, and can be altered there.

I say that it's an optional thing to change your terminal size in that Giocososo will work perfectly well without you doing it: but it is written to expect a 102x26 window and will be mostly unreadable garbage if you don't accommodate it in this way!

So, with the software installed and your terminal configured to be the 'right' size, you can run Giocososo for the first time by typing the simple command:

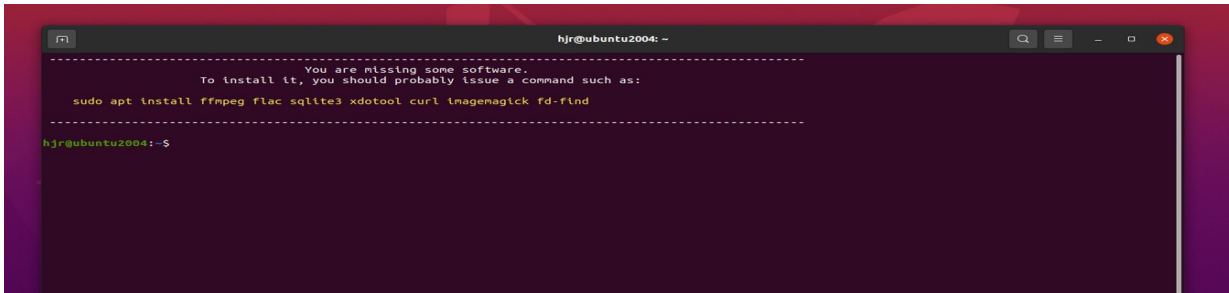
```
giocososo
```

The command to launch Giocososo *can* be much more complicated than that, as we'll go on to see. But, fundamentally, that's it: type the word 'giocososo' at the terminal prompt and the program will run in its 'default' state. Pretty straightforward, right?!

4.0 Giocosu – First Run

When first run, Giocosu will scan your system for the presence (or absence!) of necessary software prerequisites. If it finds any software missing, it will prompt you to install it, and then quit. You can then install the software using the command it suggests (or via any other software installation mechanism with which you are more comfortable), and then re-run Giocosu, until it declares itself happy with the software environment it finds itself in.

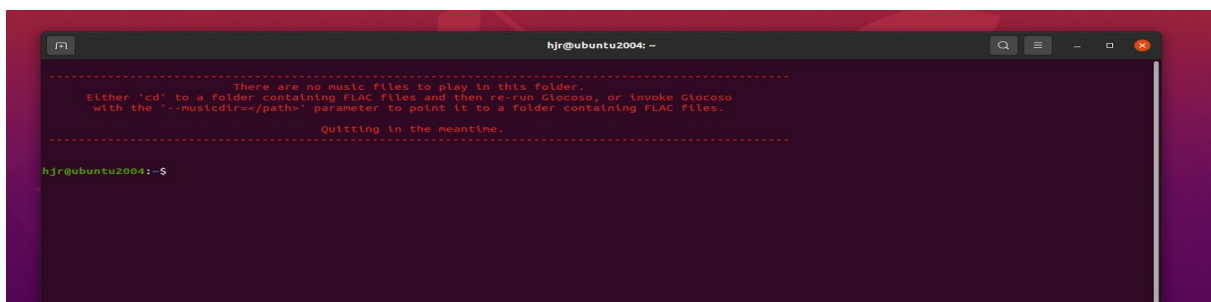
Here's a worked example on a freshly-installed (and totally default) Ubuntu 20.04.2 system:



```
hjr@ubuntu2004: ~  
-----  
You are missing some software.  
To install it, you should probably issue a command such as:  
  
sudo apt install ffmpeg flac sqlite3 xdotool curl imagemagick fd-find  
  
-----  
hjr@ubuntu2004:~$
```

You should be able to make out that the program has determined that quite a lot of new software needs to be installed, such as flac, ffmpeg, sqlite3 and so on. Notice that Giocosu is ‘distro-aware’: it knows it’s being run on Ubuntu, so it’s suggesting a **sudo apt install** command to get the required software installed. On openSUSE, it would provide a **zypper** command; on Fedora a **dnf** one, and so on. Giocosu has been tested on all major distros (anything in the top 10 of [Distrowatch](https://distrowatch.com/), basically), so it should get most of them right, but if it is being run on something out of the ordinary, it may not entirely understand how to install required packages -in which case, it’s down to you to understand your choice of distro’s package management processes!

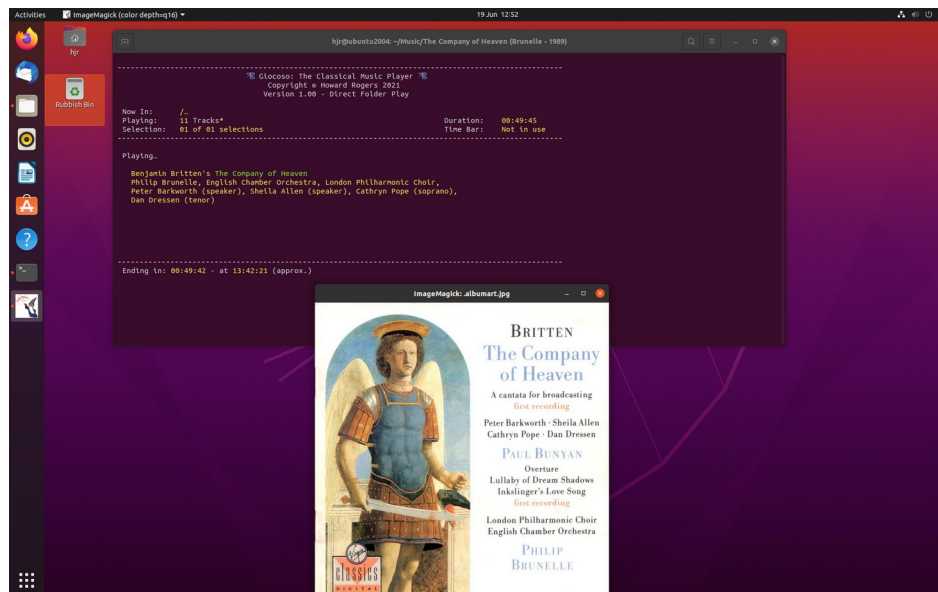
Anyway: clearly, this particular distro needs me to install some extra software, so let me run the command shown and then re-run Giocosu afterwards:



```
hjr@ubuntu2004: ~  
-----  
There are no music files to play in this folder.  
Either 'cd' to a folder containing FLAC files and then re-run Giocosu,  
or invoke Giocosu with the '--musicdir=/' parameter to point it to a folder containing FLAC files.  
  
Quitting in the meantime.  
-----  
hjr@ubuntu2004:~$
```

Well: it’s certainly a different response. This time, there’s no software missing... but Giocosu’s still not doing much! That’s because it needs to be run in a folder full of FLAC music files. If it can’t find any such files in the folder in which it’s being invoked, it produces the error message you see above: **There are no music files to play in this folder...**

So let me 'cd' to a folder I know contains some FLACs and re-run Giocososo once more:



This time, we have success: the terminal window fills up with text describing what's happening (basically, that Giocososo has found some FLAC files to play and has begun to play them) and the album art embedded in the first FLAC file found in the folder has been displayed in the middle/lower part of the screen. As you'll see in **Section 10.2.2**, the display of album art is something that can be turned off, if you prefer. The art can also be displayed smaller and larger than shown here, if its default size is a problem... but those are configuration issues for later.

For now, we can sum up installation and first run issues as:

- Download the `abc_installer` script
- Run it, supplying `--giocososo` as a parameter
- Type the command `giocososo` anywhere at the command line
- Follow the prompts to install software prerequisites
- `cd` to a folder containing some FLAC files
- Type the command `giocososo` to play them

Please note that some distros (notably, openSUSE and Debian) do not install a small program called `bc` by default. Its absence completely screws up Giocososo's ability to display messages properly. On such distros, the first run of Giocososo will prompt you to install `bc`; the second will prompt you to install any other missing prerequisites. It won't be until the *third* run attempt that Giocososo will work properly. On all other distros, it only takes two goes before Giocososo 'gets it'.

5.0 Giocosos Play Modes



Giocosos can be made to work in two quite independent ways, interchangeably, back-and-forth:

- Direct play of the contents of a specific folder or a specific FLAC file; or
- Randomised play by selecting from the contents of database

Out-of-the-box, Giocosos can only do the direct type of play. To get the randomised plays happening, you'll have to create a database and

get Giocosos to populate it by scanning your music folders. If the contents of those music folders change for any reason (such as you buying a new CD) you'll periodically need to *refresh* that database, too.

I'll describe both play modes in more detail now.

5.1 Direct Play Mode

This is the simplest way to get Giocosos to play music -and the way it's designed to work 'out of the box'. At its simplest, you open a terminal, 'cd' to a folder containing FLAC music files and type **giocosos** at the command prompt. Giocosos will then launch and play the music in the folder it's been launched from, without pause, and in file name sequence order. (So, hopefully, you've tagged your music files with correct 'track numbers' and they have been included at the beginning of the physical file names!)

For example:

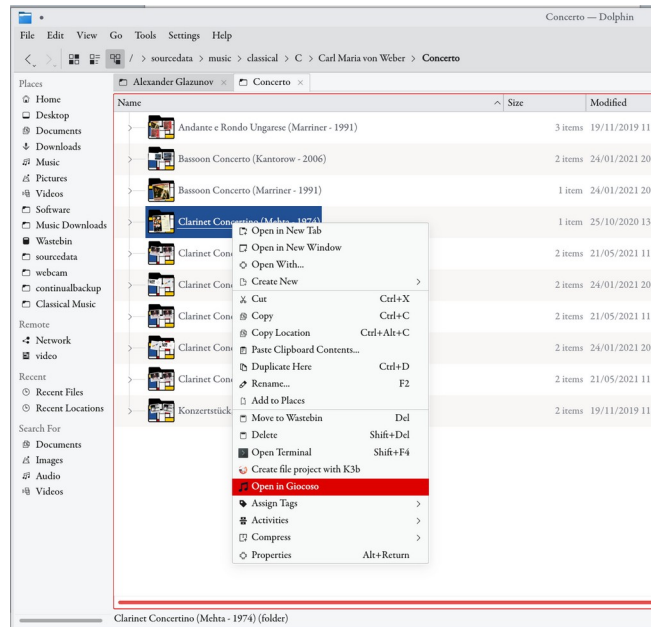
```
File Edit View Bookmarks Settings Help
New Tab Split View Left/Right Split View Top/Bottom Copy Paste Find
[hjr@britten ~]$ cd /sourcedata/music/classical/B/Béla\ Bartók/
[hjr@britten Béla Bartók]$ ls
Ballet  bela_bartok.png  Chamber  Choral  Concerto  Opera  Orchestral  Vocal
[hjr@britten Béla Bartók]$ cd Chamber/
[hjr@britten Chamber]$ ls
Piano Quintet (Szabo - 1971)
[hjr@britten Chamber]$ cd Piano\ Quintet\ \ (Szabo\ -\ 1971\)/
[hjr@britten Piano Quintet (Szabo - 1971)]$ giocosos
```

Here, I've opened a terminal and am navigating my way through my music collection folder hierarchy until I come across a specific recording I wish to play in the Béla Bartók folders. I cd into the specific folder I want, and type the plain command 'giocosos'.

When you use Giocosos for direct play of music, no record is kept of what music you've played: there is no database involved here in which to store such information. Features of Giocosos which depend on the existence of a database cannot therefore be used, either. For example, we cannot stop the playing of a particular composer's music because you've played too much of it recently (a feature called the

‘timebar’) -without a database of plays to consult, in direct play mode, we cannot know whose music has been played recently in the first place!

Note that if you ‘integrate’ Giocosio into your desktop environment (see **Section 7** for details), you can invoke Direct Play mode using a graphical file manager, such as Dolphin or Nautilus, by navigating to a folder and right-clicking the folder icon (or a specific FLAC file) and selecting ‘Open in Giocosio’ from the context menu that appears:



Here, I have navigated to a folder of music in Dolphin, right-clicked the folder name and am then able to select the ‘Open in Giocosio’ menu option, highlighted in red.

This is an exact GUI (i.e., graphical application) equivalent to `cd-ing` to that folder in a terminal session and typing the ‘giocosio’ command: it’s just that you’re using your desktop environment standard file management tools to perform the ‘play’ rather than terminal-based commands.

When Giocosio is invoked from your file manager, you are still running it in Direct Play mode, and so the same sort of restrictions pertaining to that mode still apply.

5.2 Random Play Mode

The second mode in which Giocosio can run involves consulting a database which describes your physical music collection and selecting things out of that database to play, more or less at random. You can *influence* what gets played: you can demand Giocosio only plays music by a particular composer, or recordings in which a particular conductor is taking part... but, on the whole, you leave it to the database to select randomly.

The purpose of this mode of play is really to get you listening to the dark corners of your music collection: the bits that seldom get played, because for one reason or another, they do not spring to the top of your mind if you were yourself choosing what music to play.

To invoke Giocosso in Random Play Mode, you first have to create a database; you then have to scan your music collection and have its contents recorded in the database; and then you simply run Giocosso from the command prompt with a ‘run-time parameter’ indicating which database it should consult.

You would, for example, type the following command to create a database called ‘main’:

```
giocosso --createdb --dbname=main --musicdir=/music/classical
```

The database creation process ends with an automatic scan of the /music/classical folder contents (which, depending on the size of your music collection, might take around half an hour upwards to complete).

It is important to realise that Giocosso’s database merely contains information about how to find music files on disk. The music files themselves are not in any way modified or moved by the database scanning process. The database is therefore always quite small (mere Megabytes, even if your music collection is Terabytes big and comprised of tens of thousands of files).

Once the database has been created and populated, you can invoke Giocosso most simply in random play mode like so:

```
giocosso --dbname=main
```

Notice that you always have to mention the database name if you want Giocosso to operate in randomised play mode.

The creation of a Giocosso database causes a ‘snapshot’ of your music collection to be taken. It’s the contents of this snapshot that triggers Giocosso to find FLAC files on your hard disk to play. However, if the FLAC files on your disk *change* in some way from the way that they are recorded in the database, that will cause Giocosso problems.

If you add three new CD rips to your music folders, for example... Giocosso won’t know they exist (because they didn’t exist at the time of database creation) and so cannot ever randomly select them for play. If you rename some FLACs on your hard disks: again, Giocosso won’t know that you’ve done that, and will have a record of those FLACs under their original names. If it picks one of them to play randomly, it will error... because it will try to play the file under its old name/location, not its new, and won’t understand why something it once saw in Location X as Filename Y is no longer discoverable with those details.

And so on: I won’t labour the point! If your music collection changes after you create the initial database, you need to tell Giocosso about those changes. That’s the process of **refreshing the database**, and you should do that quite frequently with a command such as:

```
giocosso --refreshdb --dbname=main --musicdir=/root-of/music/
```

Structurally, the command is very similar to the process of creating the database in the first place: we need to know the database name involved and the location of the root of your music collection folders, so we know where to scan for changes.

See **Section 9** for more details on the database creation and refresh procedures.

By default, any recording scanned into the database during a `--createdb` or `--refreshdb` operation is available to be randomly selected during a playback session where `--dbname` is in use (i.e., in Random Play Mode, everyone is equally likely to be selected for the ‘next play’).

It may happen, however, that you wish to prevent anything by a particular composer being selected for play for as long as you feel disinclined to listen to him or her. You can achieve that by ‘configuring an exclude’.

Exclusions can be configured by creating a text file called **excludes.txt** in the **\$HOME/.local/share/giocoso** folder. It has to be called that name, and it must be stored in that location, otherwise it simply doesn’t work. You simply add composers’ names to that file, one name per line. The names need to match *exactly* with the way you tagged the music files in the first place. If you tagged Symphony No. 5 as being by ‘Ludwig Beethoven’, it’s no good having a line in the excludes.txt reading ‘Ludwig van Beethoven’: if they don’t match, the exclude won’t take proper effect.

When you are happy to listen to an excluded composer again, you can either delete one line out of the excludes.txt using the text editor of your choice ...or, if he or she was the only excluded composer, you can just delete the entire file. You can always create another in the future, if the need arises.

If you have many lines in your excludes.txt, it can slow down Giocoso’s ability to randomly select something to play: it’s best to stick to no more than a couple dozen entries, really.

If you just want to prevent Giocoso picking a particular composer at random for a period of a few hours, rather than for weeks or months at a time, then you probably want to use the `--timebar` run-time parameter, rather than the excludes.txt. You could also use the `--composer` parameter with a `--negate` override to prevent the playing of a particular composer on an entirely ad hoc basis. See **Section 10.3.4** for an explanation of the timebar feature and **Sections 10.3.6** and **10.3.15** for details on selections by composer and negation.

5.3 Summary of Play Modes

To conclude this high-level overview of the two main modes of Giocoso’s operation:

- Direct Play Mode is what you get when you run Giocoso from a folder full of FLACs. There’s no database involved; there’s no randomisation of any sort. Whatever is in the folder will be played, beginning to end, in file name sequence order.
- Random Play Mode is **made possible** by the *creation* of a database. It is **performed** by mentioning the database name when running Giocoso (with the `--dbname` parameter)
- The database, once created, should be kept in synchronisation with your actual music collection by frequent *refreshes* of the database.
- The contents of the database can be partially ignored for playback purposes if an excludes.txt exists, in which specific composers are mentioned by name. Any composer mentioned in excludes.txt cannot be selected for random play, though Direct Play Mode can still be used to play them.

5.4 Pausing and Terminating Play

No matter what play mode you use, you *cannot* pause play in Giocosos. You don't pause a concerto in mid-movement at the concert hall, do you?! Right: Giocosos follows that exact same model of play. Once started, you listen until the end ☺

Of course, you can **halt** play in Giocosos at any time: press **Ctrl+C** and playback will be terminated (there will perhaps be a couple of seconds of music to play from buffered memory, but otherwise termination is pretty much instantaneous).

A piece of music that is interrupted by this Ctrl+C termination will not be recorded as having been played *at all*. It will therefore also not be scrobbled to Last.fm (assuming you're using that option). The composer of the interrupted piece will additionally not be subject to any time-bar that may be in use (see **Section 10.3.4**), since the program does not recognise partial plays as counting for anything.

Giocosos will terminate cleanly when it has played to the end the last possible 'selection' it has been asked to play (with the `--selections=y` runtime parameter: see **Section 10.3.2** for details). If you don't specify a selections parameter, Giocosos's default behaviour is to play a single piece of music through to completion and then cleanly terminate itself. All clean terminations only happen after details of the last piece of music played has been recorded in the Giocosos database and (if requested) scrobbled to Last.fm.

6.0 The Giocosso Program Display

6.1 Play Window

Whichever play mode you run Giocosso in, the main program window will always look similar to this as music playback actually takes place:

```
-----
                                %E Giocosso: The Classical Music Player %E
                                Copyright © Howard Rogers 2021
                                Version 1.06 - Randomised Play, using database MAIN
-----
Now In:      /O/Ottorino Respighi/Orchestral/Antique Airs and Dances/Hickox/Ancient Airs & Dances
Playing:     04 Tracks*
Selection:   06 of 09 selections
Duration:    00:15:14
Time Bar:    Not in use
-----

Playing...
...works of previously unplayed recordings
...with a play-length of less than 16 minutes

Ottorino Respighi's Ancient Airs & Dances, Suite I
Richard Hickox, Sinfonia 21
-----

Ending in: 00:06:47 - at 14:57:45 (approx.)
-----
```

At the top, you'll be told whether you're doing 'Direct Play' or (as in this case) 'Randomised Play via a database' -and, if a database is being used, it's name will be shown.

The folder of music being played will be listed against the 'Now In' label. This might get truncated if the path and file names concerned are very long: Giocosso uses a 101-character wide display, so tends to trim path/filenames when they exceed about 85 characters.

The 'Playing' label then displays the number of tracks found to be played. This number is invariant (that is, it doesn't change to indicate we're now playing track 2 rather than track 1, for example). It's a track *count* display, not a track *progress* indicator.

In this screenshot, we see the number of tracks suffixed with an asterisk, so that we read: **Playing: 04 Tracks***, rather than just **Playing: 04 Tracks**. The extra asterisk is there to tell you that the count of tracks is 'fake'! In other words, if you are in the habit of creating 'superFLACs', which are single physical files with an embedded cuesheet that explains where the tracks/movements 'within' the single file begin and end, then Giocosso will count the number of tracks in the cuesheet and display *that* number, rather than declaring there's just one physical file in the folder. The asterisk is, in other words, an indication of the presence of a superFLAC and the fact that the track count is 'virtual', not physical.

The 'Duration' label shows the **total** duration of the entire folder of music, not of the specific track being played, whilst the 'Selection' label indicates whether you've asked to play a number of pieces in succession and, if so, what 'play count' we're up to within that. If you run Giocosso with the parameter

--selections=9, for example, that means ‘play 9 works one after the other’, and if Giocososo was currently on the 6th of those plays, you’d see “Selection: 06 of 09 selections” displayed here.

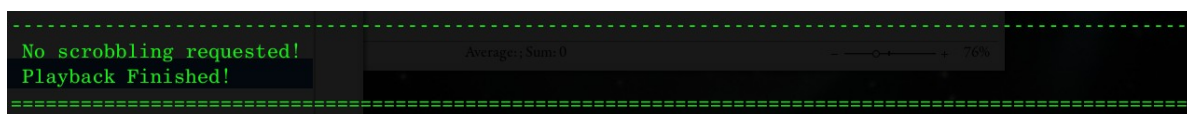
The ‘Time Bar’ label tells you how many time bar hours are in effect, if any. This can only ever work in Random Play mode: in Direct Play mode, a ‘Not in use’ message will appear instead. The timebar is a way of saying ‘if you played something by Beethoven at 9AM, don’t play anything else by him until x hours have passed’. It can only function when a database of what we’ve played and when is in use.

In the middle of the program display, you’ll see the full name of the composition that’s being played, along with the performer details. These are shown (usually) in yellow and green text and get their information from the ARTIST, ALBUM and COMMENT tags.

If you have applied any ‘selection’ run-time parameters (such as ‘play something by composer X’ or ‘play a piece of music belonging to genre Y’ or ‘play something where singer Z is performing’), then these ‘filters’ will be displayed in blue text, so you can see something of the reason why Giocososo has decided to play a piece. In the example screenshot from earlier, for example, you can see I’ve asked to play previously unplayed works (that would be because I used the **--unplayed** runtime parameter) and also asked only to play works that last less than 5 minutes (because I used the **--maxduration=5** runtime parameter).

At the very bottom of the main program display, you will see an ‘elapsed time’ count ticking down to the estimated conclusion time (so you know when the currently-playing piece of music will stop playing). The countdown is for the currently-playing composition. If you’ve asked to play 6 different selections, for example, the ‘end time’ shown will be for each selection in turn, not when the entire set of 6 compositions has been played. That’s partly because Giocososo doesn’t *know* when the complete set of selections will end: it hasn’t made them yet, so cannot possibly know their various individual durations!

This area of the screen will also show messages about when a ‘play’ has been scrobbled to Last.fm (if you’ve asked for that to happen) or a ‘No scrobbling requested’ message (if you haven’t) and so on. When a complete cycle of play ‘selections’ has been made, you’ll be told here, too, that ‘Playback has finished’:

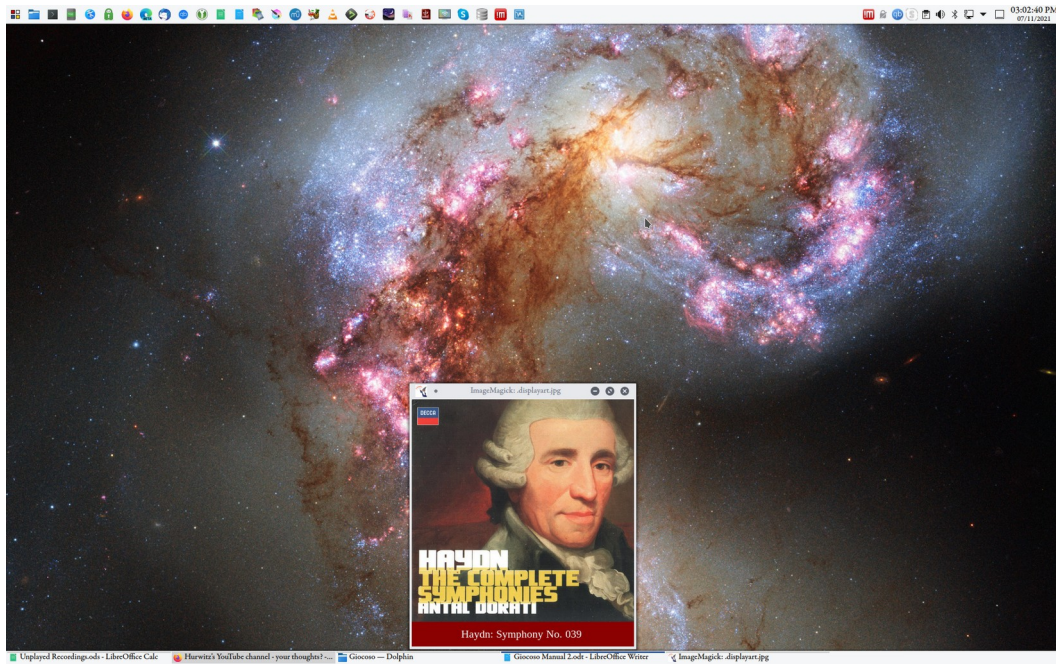


As you see, the program display is (deliberately!) sparse: it gives you information about what choices you’ve made and the randomising choices Giocososo has made. It tells you when you can expect playback to end. It does *not* indicate per-track information, ever: classical music listeners should be listening to ‘compositions’ not ‘tracks’!

6.2 Album Art Display

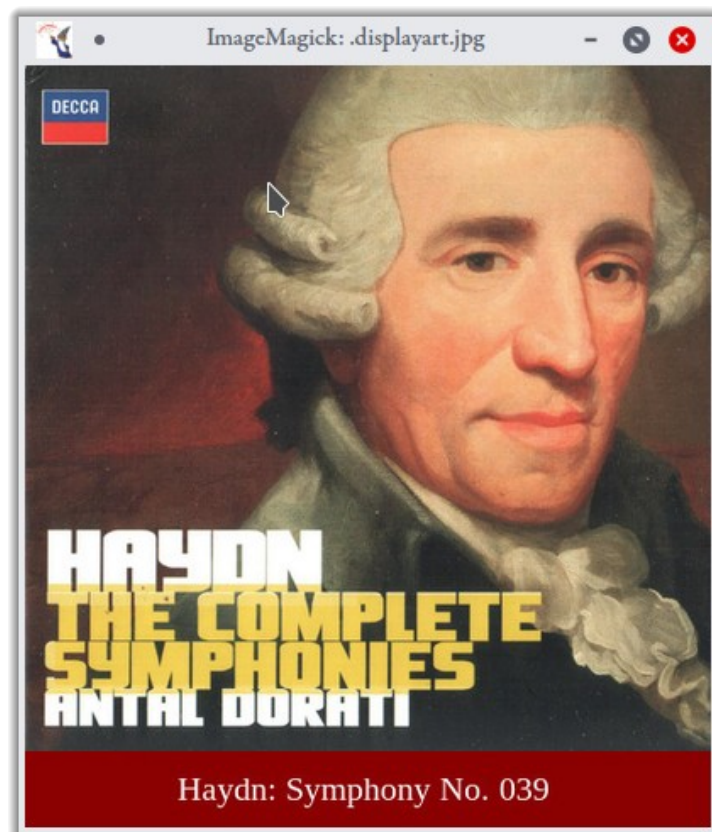
By default, whenever Giocososo starts playing something in its main play window, it will also extract the album art from the first FLAC found in the folder selected for play and display that (usually in the middle-lower part of your screen) as a more visual clue to what’s been selected.

The general effect ends up looking something like this, therefore:



The default size for the album art window is 450x500 pixels. A 450x450 pixel window displays the album art proper, but an extra 450x50 bit is bolted on underneath that, in which the composer/composition names are displayed. The result is the 450x500 pixel display shown here.

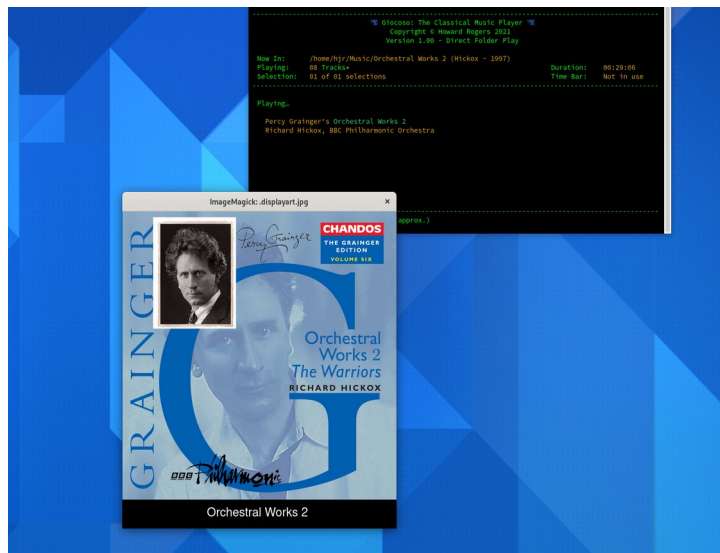
Here's a perhaps clearer picture of the effect of displaying album art plus the composition name:



If a 450x500 window is too large or too small for your monitor, you can decrease or increase its size with the `--artsize` parameter. See **Section 10.2.2** for details, but in summary: `--artsize=large` gives you a 900x960 art display window; `--artsize=medium` gives you the default 450x500 one; and `--artsize=small` yields a 300x360 one. Finally, if you'd rather not display album art at all, then `--artsize=none` will switch the window off completely.

The text in the bottom 'bar' of the album art display is derived by taking the last name of the COMPOSER tag and as much of the ALBUM tag as is thought will fit the available space. Note that this causes some composers to be displayed a tad awkwardly: Benjamin Britten will be perfectly fine displayed as 'Britten', but Ralph Vaughan Williams will be displayed as plain 'Williams' and Sir John Blackwood McEwan will be shown as just 'McEwan'. There's not a lot that can be done about this, I'm afraid.

Note that the text in the 'composition name' label is always "Liberation Serif", which is an open source font commonly installed on all distros with which I'm familiar. If you don't have it installed, though, then the display results may be unpredictable. Here, for example, is what happens on Arch (which is famously installable in very minimalist mode, so extra fonts don't just 'happen' by accident!):



As you can see, the caption is still being displayed, despite Liberation Serif **not** being present on this particular installation of Arch. You'll maybe note, however, that it's being displayed in a font which is clearly a **sans serif** of some kind (that is, there are no tail or strokes finishing off the shape of each letter). On this occasion, therefore, Giocosso has been able to silently substitute in an alternative font. It *should* be able to do the same whenever Liberation Serif is missing... but results may vary!

As an aside, the Liberation fonts are installable easily enough in most cases by using your distro's standard package manager. If all else fails, they're [available for free download from Github](#). But as you've just seen, you may well not need them at all, if Giocosso can work around their absence.

Anyway: the album art is there to display if you want it, by way of a quick visual clue to what's playing and with a short caption that gets more specific about it, too.

7.0 Integrating Giocosos with the Desktop

As previously described, Giocosos at its simplest can be invoked from a terminal session in a folder containing FLACs in what is known as ‘Direct Play Mode’. Giocosos can also be invoked in Direct Play Mode via the operating system’s standard File Manager: one can then navigate to a folder using a familiar GUI tool, right-click the folder (or FLAC file) and ask for it to be opened in Giocosos. Having the File Manager offer that option, however, requires ‘integrating’ Giocosos with the desktop environment, as a one-time operation.

To do that, you need to run this command:

```
giocosos --integrate
```

You will need to be able to provide a sudo password at one point, because integration requires writing files to /usr/bin, which ordinarily requires root permissions. Some desktop environments will respond to the integration process immediately. In KDE, for example, you’ll immediately be able to right-click a folder full of FLACs and select the ‘Play in Giocosos’ right-click menu option. Gnome similarly responds immediately. In XFCE, however, it’s necessary to log out and log back in before the effects of desktop integration become visible.

Running that command is really all you need to do, but I’ll now document what the Giocosos ‘integrate’ process actually does (or tries to do!) to your operating system to make things work, so if necessary you can go through the various files and make manual changes as appropriate, should things not appear to be working for some reason.

7.1 KDE - Plasma

The Giocosos integrate option creates a KDE ‘Service Menu’ (which is KDE’s term for ‘a menu option that appears when you right-click a folder or file’). It does this by creating a new file called **giocosos.desktop** inside the folder **\$HOME/.local/share/kservices5/ServiceMenus**. The file should contain the following:

```
[Desktop Entry]
Actions=PlayinGiocosos
MimeType=inode/directory
Name=Play in Giocosos
ServiceTypes=KonqPopupMenu/Plugin
Type=Service
X-KDE-Priority=TopLevel
Terminal=True

[Desktop Action PlayinGiocosos]
Name=Play in Giocosos
Icon=/usr/share/applications/giocosos.svg
Exec=/usr/bin/giolauncher.sh
```

Note the MimeType entry: it means you can only right-click *folders*, not individual FLAC files.

This now gives you the ability to right-click a folder and have any FLACs it (directly) contains played in Giocosio -but, clearly, when you right-click in a GUI, you can't also supply some command-line parameters that might be important. For example: you can't say to 'scrobble' this play; or to direct this play's audio to hardware device X or Y. By default, therefore, Giocosio plays initiated by right-clicking a folder in Dolphin will use the hardware device called "default" and will not scrobble.

To allow you to change these defaults, the Giocosio integration option also created a folder and script called `$HOME/.config/plasma-workspace/env/path.sh`, in which it added these lines:

```
#!/bin/bash
export GIO_DEVICE="default"
export GIO_SCROBBLE=0
```

Replace the values for each parameter shown as appropriate. If you want scrobbling to happen when you right-click a folder to play its contents, set `GIO_SCROBBLE=1`. And replace 'default' for the `GIO_DEVICE` setting with whatever hardware device ID you have determined works for your system.

Be warned that the parameters in the `path.sh` will apply to Giocosio plays that do *not* occur as a result of right-clicking a folder, too. The `plasma-workspace/path.sh` file technique is not perhaps the obvious place where you'd set environment variables (one tends to think of `.bashrc`, for example, as the more common alternative), but it's the only way to get GUI KDE apps to become aware of environment variable settings.

7.2 XFCE - Thunar

When running under XFCE, the Giocosio integrate option creates a new 'Custom Action' (which is what XFCE's file manager, Thunar, calls the items in the menu that appears when you right-click a file or folder). It does this by appending text to the `$HOME/.config/Thunar/uca.xml`, as follows:

```
<action>
<icon>/usr/share/applications/giocosio.svg</icon>
<name>Play in Giocosio</name>
<unique-id>1624366500142809-1</unique-id>
<command>/usr/bin/giolauncher.sh %f</command>
<description>Play in Giocosio</description>
<patterns>*</patterns>
<startup-notify/>
<directories/>
</action>
```

This new menu option does not, however, take effect until you log out and log back in.

7.3 Gnome - Nautilus

Under Gnome, the `--integrate` option creates the usual `/usr/bin/giolauncher.sh` "stub" script and then simply copies it to `$HOME/.local/share/nautilus/scripts`. This is simple -but also results in a less-attractive right-click option: instead of being able to click a 'play in Giocosio' option when you right-click a folder of FLACs, the best you can do is right-click, select 'Scripts', and then click on `giolauncher.sh`. It works, but it's not particularly pretty, I'm afraid.

7.4 Budgie

Budgie is functionally identical to Gnome, so Giocoso's integrate option on that desktop also creates a shell script in the `$HOME/.local/share/nautilus/scripts` folder, and similarly creates an immediate opportunity to right-click a folder of FLACs and select *Scripts* → *giolauncher.sh* to get it to play in Giocoso.

7.5 Other Desktop Environments

I regret to say that if your choice of desktop environment is not one of KDE, Gnome, XFCE or Budgie... Giocoso won't automatically integrate for you. Hopefully by reading the above, you'll have an idea of what you'll need to do *manually* to get right-click-and-play-in-Giocoso functionality for your particular choice of desktop environment, though. Basically, all that's required is to get your file manager to invoke a 'giolauncher.sh' script, which in turn calls the giocoso.sh script proper... but, unfortunately, Giocoso cannot automate this process for you at the present time.

7.6 Remembering Album Art Display Location

If Giocoso is asked to display album art (with the `--artsize` parameter set to large, medium or small), then it will always *try* to display it touching the bottom most part of your screen display, in the middle. Usually however it cannot *open* the album art there: desktop environments of all description tend to like opening things in the top-right of their screens. So, you will tend to see a sort-of 'flash' effect as the album art window is moved to its lower-middle location. That can be slightly unnerving to watch... and sometimes it doesn't work anyway! If other application windows are in the way, the movement algorithm cannot always do its work.

On KDE, it's possible to fix this by getting KDE itself to remember where the album art display window should be opened and located *every time* -and this works every time without fail. Other desktop environments may or may not have such a feature: there was a Gnome extension that allowed it on that desktop, for example, but it's no longer compatible and no replacement has yet been made that I'm aware of.

So, just for KDE users for now: once you've got some album art displayed in the correct location, right-click it's top bar and select **More Actions** → **Configure Special Application Settings** from the context menu that appears.

Now click the **Detect Window Properties** button: your mouse cursor will turn into a '+' or crosshair pointer. Click on the album art display window with it and you'll see a panel appear called '**Add property to the rule**'. Various items will appear describing the properties of the window you just clicked on.

The two properties we care about are the window's **title** (which will always be *ImageMagick: .displayart.jpg*) and its **position**, which will display with its current x and y coordinates relative to your screen size as a whole, in brackets. On my 4K monitor, for example, I see it displayed as (1,620 1,464) or similar. That means the window starts 1620 pixels across (which is approximately half of my 3840x2160 monitor settings) and 1464 pixels down (which is toward the bottom of my screen, since 0, 0 is the top left-hand corner of your screen).

If you hover over a property, you'll see a '+' button appear on the far-right of it. For the title and position properties, therefore, click their respective plus signs to add them to as new property rules and then click the 'X' button to close the 'Add property to the rule' window. You'll now be left looking at something like this:

Window matching

Description Application settings for display

Window class (application) Exact Match display

Match whole window class Yes No

Window types All selected

Window title Unimportant geMagick: .displayart.jpg

Size & Position

Position Apply Initially 1,620 x 1,464

+ Add Property... Detect Window Properties Instantly

OK Cancel

The main thing to do now is to click on that 'Unimportant' drop-down box next to the 'Window title' label: select **Exact match** instead.

The coordinates for the 'Position' rule are those derived from the current location of your album art display window: if you're happy with *its* placement, you'll probably be happy with the placements of identically-named windows in the exact same location in the future, so you can probably just accept whatever you see displayed here. Very roughly, the first number should be half-way across your screen, less half your --artsize setting; and the second number should be the vertical resolution of your screen, less your --artsize plus 60 pixels for the caption border.

Anyway: once you've got a title+Exact Match and a position rule in place with numbers you're happy with, click the **OK** button to save the new settings.

You'll now find that any time Giocosio starts playing a new piece of music, the album art will immediately appear in the correct location ...and there won't be any unnerving windows flying about your screen to get it positioned appropriately! Sadly, this sort of application positioning is not available, as far as I know, for anything other than KDE: there's a reason I use it, I guess!

8.0 Scrobbling

“Scrobbling” is the slightly odd word that means ‘send details of the things to which I’ve been listening up to a cloud service’ -and the ‘cloud service’ everyone usually thinks of in this regard is [Last.fm](https://www.last.fm/). It can be a useful thing to do, if over years rather than months, you allow it to build up a long-term history of your listening habits.

That said, Last.fm is infested with people who wouldn’t have the first clue about classical music! The way scrobbles are usually captured and recorded is also clearly not informed by classical music practice: everything is track-based, so if you listen to Beethoven’s 5th Symphony, that counts as 4 scrobbles, one per ‘track’ (i.e., per movement). Despite that, I’ve been recording my plays to Last.fm since 2008, and it is nice to discover [who my top 10 composers are and so on](#).

These days, the track-centric approach to recording ‘listens’ annoys me more than it did in 2008!

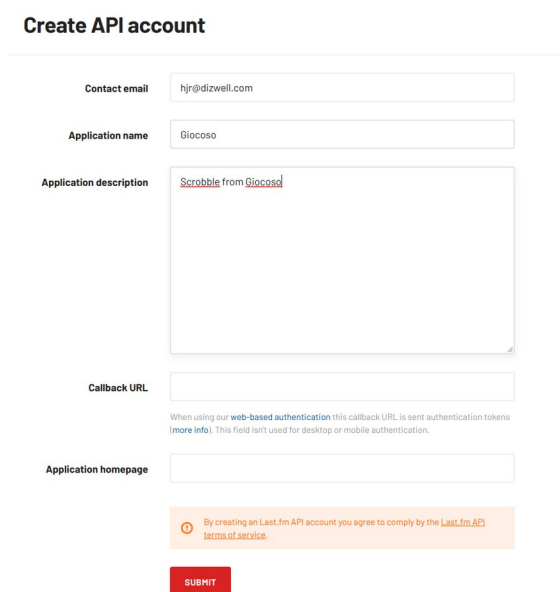
Thus: Giocosos **does not scrobble by default**, though it can do so if you do some fairly painless one-off configuration, and I’ll describe that process now.

8.1 Configuring Scobbling

To start with, you will first need to [create an account at Last.fm](https://www.last.fm/api/account/create) (which is free to do, but is completely independent of anything to do with Giocosos, so don’t complain to me about them, please!) Once you have registered as a new user, you can log on to your account for the first time. Once you’ve done that, you need to do something which Last.fm call ‘creating an API account’... which sounds complicated, but really means ‘set up some security credentials so that third-parties can record ‘listens’ into my Last.fm account.

So, to create these credentials, you have to be logged in to Last.fm and then visit <https://www.last.fm/api/account/create>.

When you do so, you’ll see this sort of screen:



The screenshot shows the 'Create API account' form on the Last.fm website. The form has the following fields and content:

- Contact email:** hjr@dizwell.com
- Application name:** Giocosos
- Application description:** Scrobble from Giocosos
- Callback URL:** (empty field)
- Application homepage:** (empty field)

Below the fields, there is a small text block: "When using our web-based authentication this callback URL is sent authentication tokens (more info). This field isn't used for desktop or mobile authentication."

At the bottom, there is a red 'SUBMIT' button. Above the button, there is a small orange box with a warning icon and the text: "By creating an Last.fm API account you agree to comply by the [Last.fm API terms of service](#)."

You need to fill the screen as you see me doing here: the contact email will be pre-filled for you, so you just need to supply something for the 'Application name' and 'Application description'. What you supply doesn't have to be very descriptive and the suggestions you see here will work fine. You should definitely leave the 'Callback URL' and 'Application homepage' fields empty.

Click the red 'Submit' button when you're ready, and the screen will change to this:

API account created

Here are the details of your new API account.

Application name	Giocososo
API key	7f154d44...b9075
Shared secret	7faa68...e47a49
Registered to	dizwell

I've blanked out parts of the 'API key' and 'Shared secret', hopefully for obvious reasons! These allow third parties to post to your Last.fm account, so it's important that you don't share them around lightly!

You will need both pieces of information in just a moment, however, so keep the web browser page handy, or copy-and-paste the details shown into a text editor or equivalent, so you can access the details when necessary.

We now switch away from Last.fm and their security shenanigans and instead run Giocososo on the home PC once more ...but this time, using a new command line switch to make it do Scrobbling configuration. The command you want to run at a command line prompt (in any folder you like: it doesn't need to contain music at this stage) is:

```
giocososo --scrobble-config
```

This will cause Giocososo to display a welcome message, prompting you to press a key to continue (or Ctrl+C to quit). I'll assume you press any key at all to keep going!

Giocososo doesn't waste any time: it immediately prompts you to supply the 'Last.fm API Key':

```
-----
  [i] Giocososo: The Classical Music Player [i]
    Copyright © Howard Rogers 2021
    Version 1.00 - Non-Play Activities
  -----

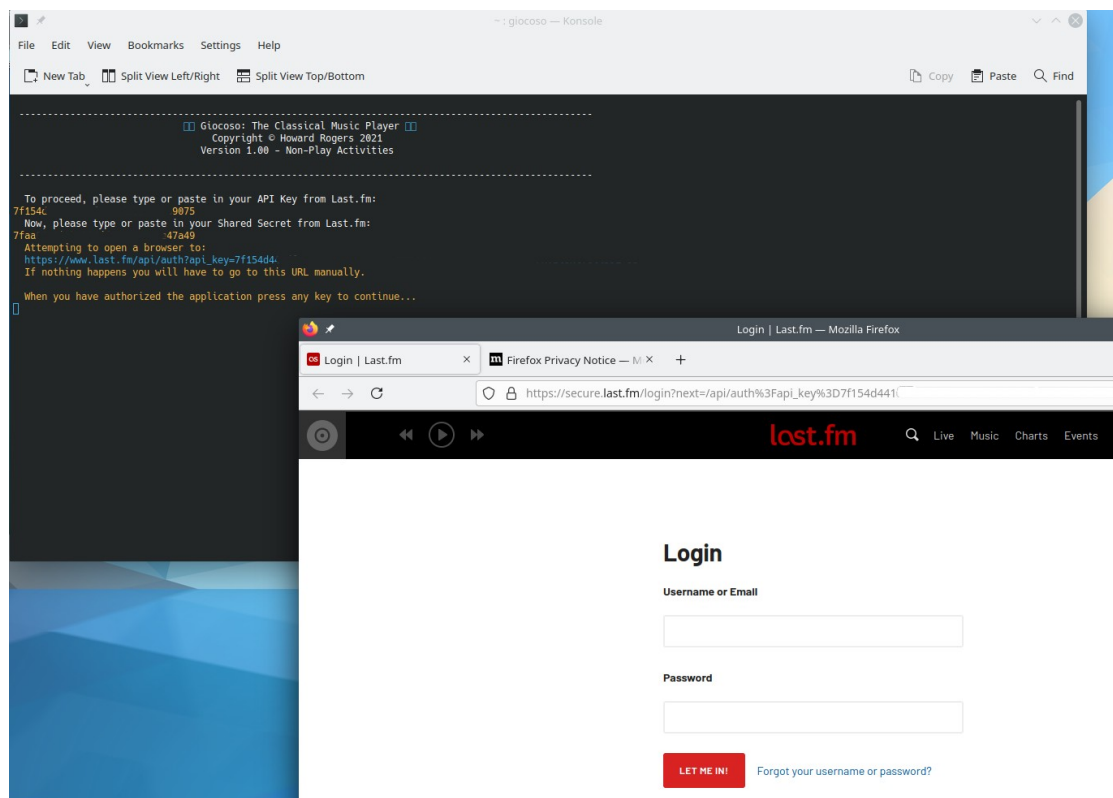
To proceed, please type or paste in your API Key from Last.fm:
[ ]
```

So, you just copy-and-paste in the information that was previously displayed on the Last.fm API Account screen. In my case, I'll paste in the '7f154d4..' key. Press Enter to submit it and move on:

```
-----  
    [ ] Giocososo: The Classical Music Player [ ]  
        Copyright © Howard Rogers 2021  
        Version 1.00 - Non-Play Activities  
  
-----  
  
To proceed, please type or paste in your API Key from Last.fm:  
7f15c6e9d89075  
Now, please type or paste in your Shared Secret from Last.fm:  
[ ]
```

Giocosio now prompts you to supply the ‘Shared Secret’ from that same Last.fm screen we saw before, and again you copy-and-paste that in, so you get the data entry exactly right. Press Enter to submit it.

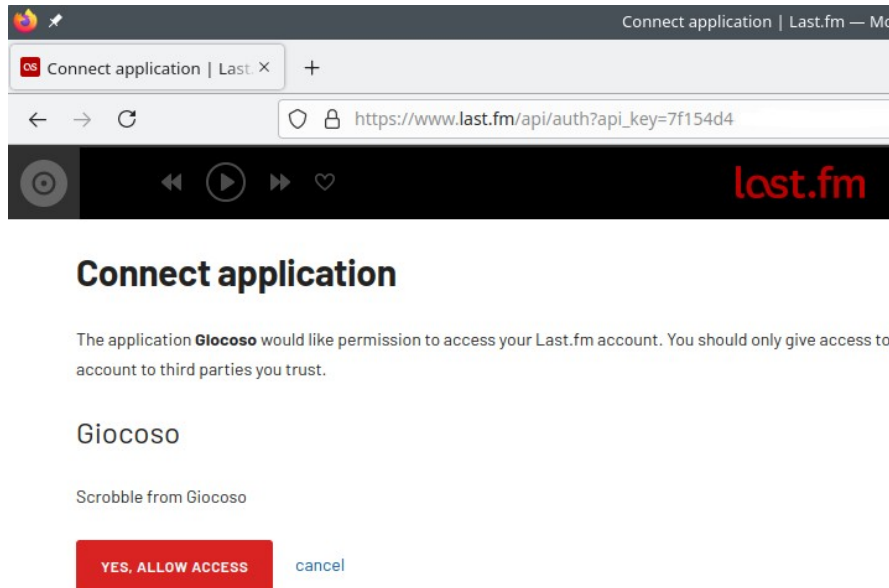
It's quite hard to describe what now happens, because it should all happen automatically and all sorts of things happen seemingly simultaneous! But broadly speaking, you should end up looking at a screen that looks a bit like this:



The script running in the background has taken the two pieces of API security information (which I've again mostly blanked out) and used it to construct a URL (in blue). It then attempts to open your default web browser at that new URL. Most often, it will succeed in doing this -but occasionally, the browser window will either pop-*under* your other windows, so you won't know that it has worked; and

sometimes, it simply doesn't work at all! If that happens, you just need to copy the URL in blue and *manually* open a browser and paste that URL into the address bar.

However you do it, you should end up at a Last.fm web page asking you to log into your account. You just provide your standard Last.fm username and password credentials to get to this screen:



Here, you are being asked to authorise Giocoso to be able to scrobble to your Last.fm account. You should click the 'Yes Allow Access' button... and you'll receive a bright green confirmation message saying that 'you have granted permission to Giocoso to use your Last.fm account'. At this point, switch back to Giocoso running on your PC and press Enter to confirm that you've authorised Giocoso to scrobble -and Giocoso will terminate with a confirmation that the scrobbling configuration data has been successfully stored.

Incidentally, if you ever want to revoke Giocoso's permission to scrobble Music, it's easy to do: just click your personal icon in the top-right of the Last.fm web page (after you've logged in), and click *Settings* → *Applications*. Just click the 'Disconnect' button against an application to de-authorise it.

Anyway: that's the scrobble configuration from Giocoso's point of view now complete. Giocoso can now scrobble correctly.

8.2 Getting Giocoso to Scrobble

Remember that Giocoso *doesn't scrobble by default*, even when it's been authorised to do so. You have to manually tell Giocoso to do so, every time you run it. The command would be:

```
giocoso --scrobble
```

...so it's not exactly difficult! If you forget to supply that extra parameter, however, you'll see this sort of message at the end of a 'play' of a composition:

```
-----
Giocosos: The Classical Music Player
Copyright © Howard Rogers 2021
Version 1.00 - Direct Folder Play

Now In:      /_Concerto (Françaix - 1973)/Double Piano Concerto (Matiakh - 2014)
Playing:     04 Tracks*
Selection:   01 of 01 selections
Duration:    00:29:07
Time Bar:    Not in use
-----

Playing...

Jean Françaix's Double Piano Concerto
Ariane Matiakh, Deutsche Staatsphilharmonie Rheinland-Pfalz, Mona and Rica Bard
(pianos)

-----

No scrobbling requested!
Playback Finished!
=====

hjr@localhost:~/Music/Piano Concerto (Françaix - 1973)/Double Piano Concerto (Matiakh - 2014)> |
```

The ‘no scrobbling requested’ message is the clue that, though scrobbling has been configured, it wasn’t specified at the time I launched this particular ‘run’ of Giocosos.

(If you instead get a ‘Scrobbling not configured’ message at the end of a ‘play’, it means Giocosos doesn’t think you’ve ever run it with the **--scrobble-config** parameter successfully. You need to have success with that before any attempt at scrobbling can be made).

If you *do* manage to remember to add the **--scrobble** parameter to your command line when running Giocosos, you’ll see this end-of-play message instead:

```
-----
Scrobbling Complete!
Playback Finished!
=====
```

If remembering that **--scrobble** switch every time is too awkward, though, you can set an environment variable, called **GIO_SCROBBLE** to a value of 1 in your **.bashrc** or whatever other file you use to set your session environment variables. Here, for example, is a snippet from my own **.bashrc**:

```
GNU nano 5.7 .bashrc

GIO_DEVICE="plughw:2,0"
GIO_SCROBBLE=1

alias splitflac='cao --asunder -x'
alias splitape='cao --asunder -x'
alias apetooflac='auac -i=ape -o=flac'
alias ripsacd='/usr/bin/sacd_extract -i 192.168.137.40:2002 -s -z -2 -o /home/hjr/Music'
alias ccdt='ccdt --namebits --namereplace'

export EDITOR=nano
```

If GIO_SCROBBLE is set to 1, then Giocosos will automatically scrobble, even if the **--scrobble** parameter is absent when it is launched. Setting that environment variable to 0 is the same as not setting it at all: Giocosos reverts to its default behaviour of not scrobbling unless explicitly told to do so on the command line when running the program.

In case you're wondering: if you set GIO_SCROBBLE=0 (i.e., 'no scrobbling, please') and then run Giocosos with the **--scrobble** parameter ...the run-time parameter 'wins' and Giocosos *will* scrobble. The explicit run-time parameters always take precedence over generally-set environment variables.

8.3 How Giocosos Scobbles

I want to end by mentioning that, once you've configured Last.fm to let Giocosos scrobble, and remembered to tell Giocosos to scrobble... *how* it scobbles is going to be rather unlike any other media player on Linux that I know of!

Remember that Last.fm is 'track-centric', where Giocosos thinks in 'whole composition' terms? Well, that difference shows when it comes to scrobbling!

If you play a 3-movement piano concerto, Giocosos won't scrobble *anything at all* until the entire concerto has been played to its conclusion. When the last bar of the last movement fades away, that's when Giocosos will scrobble *all three tracks of the work at once*. If you therefore interrupt a Giocosos 'play' half-way through (by pressing Ctrl+C), then *nothing* of that play will get scrobbed at all.

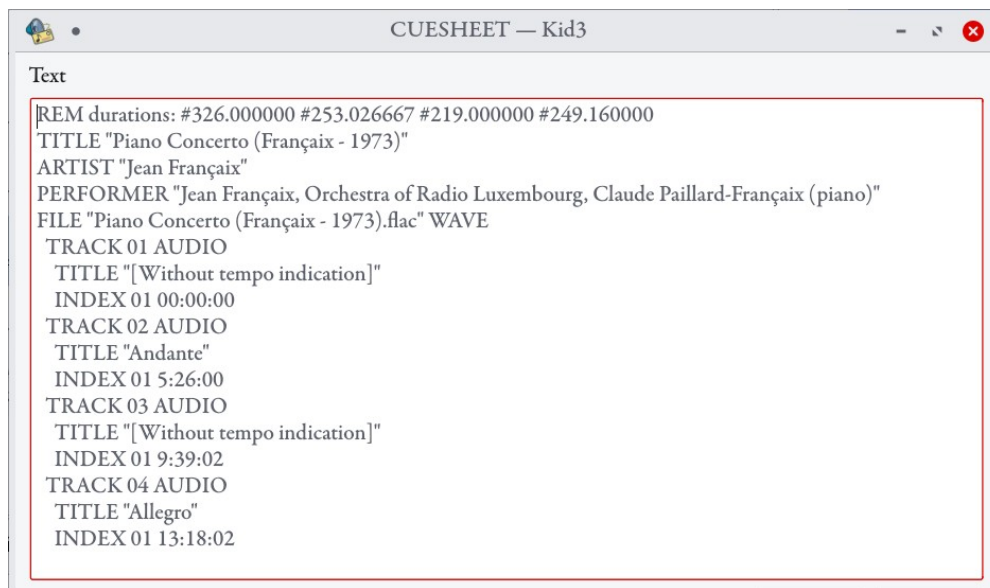
Essentially, Giocosos takes the view that if you walk out of a symphony half-way through, you haven't actually heard the entire symphony, so it shouldn't record you as having listened to *any* of it! Classical music doesn't work that way... and neither should you!

If you run Giocosos with the **--selections=x** parameter, so that Giocosos plays *x* compositions in sequence, one after the other, then Giocosos *will* scrobble each selection as it ends. It doesn't wait until the entire cycle of selections is complete before scrobbling things, in other words.

8.4 Scrobbling with Embedded Cuesheets

This may not apply to you, but some people (me included!) prefer to take the four movements of a Beethoven symphony (say) and combine them into a single FLAC track -usually referred to in these parts as a 'superFLAC'. All four movements are present 'within' the single FLAC file *logically*, but it's physically a single file. Usually when you combine individual track FLACs into a superFLAC in this way, you'd also embed a piece of information called a 'cuesheet' into the FLAC file at the same time. The cuesheet 'describes' the superFLAC, indicating when its constituent 'tracks' begin and end.

For example, here's a cuesheet explaining what's going on 'inside' a single FLAC that is the entire Françaix piano concerto:



From this cuesheet, we can tell that the third movement begins 9 minutes and 39.02 seconds in from the beginning of the piece. The fourth movement starts at 13-and-a-bit minutes in (so we can deduce that the third movement lasts for around 3~4 minutes) and so on.

Anyway: the point is, if you get Giocosio to play this composition, it will play it without a problem... but what should get scrobbled? A single record saying 'he played the Françaix piano concerto'? Or four scrobbles, saying 'he played the first movement, then the second, then the third and finally the fourth'?

Well, Giocosio recognises that Last.fm is a track-centric website, so it *will* scrobble the four individual movements, rather than just the one concerto -even though it's only one physical file on disk. But as I've already mentioned, Giocosio won't scrobble *anything at all* until all four movements have been played. So it's a sort of compromise: yes, Giocosio acknowledges the "trackiness" of works, but no, it won't ever allow you to scrobble only parts of a composition!

In the specific case of this piano concerto, for example, this is what the Last.fm website for my account shows -and as you can see, it displays 4 'tracks', even though they're 4 *virtual* tracks, derived from the embedded cuesheet, rather than 'real' ones derived from the existence of separate physical files :

Recent Tracks				⚙
	♥	Allegro fieramente	Jean Françaix	2 minutes ago
	♥	Scherzando	Jean Françaix	2 minutes ago
	♥	Allegro giocoso	Jean Françaix	2 minutes ago
	♥	Doppio più lento	Jean Françaix	2 minutes ago
	♥	Drommen om 'Glade Jul'	Carl Nielsen	an hour ago
	♥	Overture Šárka	Leoš Janáček	an hour ago

You may notice that because Giocososo ‘delays’ scrobbling until the end of the work, the entire set of ‘parts’ of the work all appear to be scrobbed at the same time: the Françaix concerto lasts 29 minutes, after all, so there is no way that all four movements could have *all* been **played** ‘2 minutes ago’ in real life! But this is just what happens when Giocososo stores up all scrobbs for a work until the entire work is complete: the scrobble time for all tracks (or ‘virtual tracks’, if we’re reading an embedded cuesheet) is set to be the time when the *last* of them finished playing.

Summing up, then:

- Giocososo needs to be configured to scrobble (--scrobble-config)
- Giocososo needs to be instructed to scrobble (--scrobble)
- Giocososo will only scrobble when the playback of composition ends
- Giocososo will scrobble multiple tracks if your music is physically stored as one-track-per-movement
- Giocososo will scrobble multiple tracks even if your music is physically stored as one-track-per-composition, provided that it can find an embedded cuesheet that describes multiple movements
- When multiple tracks are scrobbed, the scrobble time-stamp for all of them will be identical

9.0 The Giocosso Database

Giocosso main reason for existence is its ability to randomise playback of parts of a large music collection -but it can only do so if run with a database backend (otherwise, it just dumbly plays the contents of a folder you point it at).

- That database has to be created with the **--createdb** run-time parameter.
- It is then kept up-to-date with changes in your music collection with the **--refreshdb** parameter.
- It is thereafter used by Giocosso any time the program is run with the **--dbname** parameter.

It's important to realise that creating a database doesn't mean moving your physical music files anywhere or altering them in any way. The database is a store of information as to where, on disk, your music files can be found: it does not contain the music files themselves. Accordingly, the database is small and compact, even after a very large music collection has been scanned: my 1.7TB music collection is represented by a Giocosso database that is only 11MB big, for example.

The Giocosso database is a sqlite3 one (not MySQL, in case anyone was wondering). Sqlite3 is an open source, compact, highly portable and performant relational database.

9.1 Creating a Database

To create a database in the first place, you must run Giocosso with the following style of command;

```
giocosso --createdb=mymusic --musicdir=/path/to/root/of/music-collection
```

The 'createdb' parameter takes as its value the name of the database you want to create. If you fail to supply a name (i.e., if you just say "--createdb", with no equals sign or parameter value), then a database called 'music' will be created by default.

If the database name you supply already exists, you'll be warned about this and told that no database creation is possible and that you should probably instead be trying to 'refresh' the database which already exists.

There is no limit to the number of databases you can create, though they must each be uniquely named, and only one can be used for playback at a time. If you had classical and jazz music collections, for example, and wished to keep them distinct, you could do two 'createdb' runs, naming each database something like 'classical' and 'jazz', and you'd thereafter be able to play each type of music distinctly with different play-runs of Giocosso using different --dbname parameters.

When you ask to create a database, you are required to supply a --musicdir parameter and will be warned if you don't. This is so that, once the database is created physically, the music files stored within the supplied 'musicdir' can be scanned and have their metadata tags read for inclusion into the new database's tables. This scanning process can take quite a long time to start, so be patient. Once it has started, however, you will be able to see the progress of the scan in the upper part of the program display.

The `musicdir` parameter should point to the ‘root’ of your folder tree within which your digital music collection can be found. There is no practical limit to the ‘folder depth’ that Giocososo will descend to in order to find FLAC files. In my own case, for example, one of my Beethoven symphonies can be found in the following folder:

```
/sourcedata/music/classical/L/Ludwig van Beethoven/Symphonic/Jochum (LSO) Cycle/Symphony No. 3 (Jochum - 1976)/
```

...so you can see that’s quite a few folders ‘deep’: but the root of all my music is simply:

```
/sourcedata/music/classical
```

...and that’s the folder I’d add as the value for the `--musicdir` parameter, and the symphony will be ‘discovered’ within that path perfectly well.

Should your desired `musicdir` path contain spaces, please wrap the entire thing in double quotation marks. For example:

```
giocososo --createdb=music --musicdir="/my music/classical music"
```

Ideally, you wouldn’t have spaces in the ‘root’ of your music folder path like that, but if you insist on it, Giocososo can at least deal with it, double quotes permitting.

9.2 Refreshing a Database

If your music collection is very static, you may never need to refresh the Giocososo database -but if you acquire new recordings over time, or alter the metadata associated with existing ones, then you need to keep the Giocososo database ‘in synch’ with the evolving state of your music collection by means of performing a database refresh.

In a database refresh, the contents of the two principal tables are completely wiped and re-populated from scratch by performing a brand new scan of your music folder and its sub-folders. There is no ‘differential’ scan with Giocososo: it does **not** do a quick scan whereby it detects only the changes since the last scan, for example. It always wipes the database (but **not** your play history) and starts from scratch.

A specific consequence of this way of working is that any ID numbers assigned to a particular recording are likely to change over time: they are not invariant between refreshes.

Since a database refresh closely resembles a database (re)creation, the command syntax for performing one is very similar to the way you create a database in the first place:

```
giocososo --refreshdb --dbname=music --musicdir=/path/to/music-collection
```

Again, it can take several minutes for the scan to begin when dealing with a large music collection, but eventually the program will show you what folders it is processing so you get some idea of where it’s up to and how much work there is still to do before completion.

Music files which exist on disk but are not incorporated into the Giocososo database cannot be played by Giocososo in Random Play Mode -but it is always open to you to visit the files concerned in your distro’s file manager and play them in Direct Play Mode.

There is no hard-and-fast rule about how often to refresh your database: as mentioned, it really depends on how ‘dynamic’ your music collection is. If you are acquiring half a dozen new recordings a week, say, then maybe a weekly refresh might be in order; if it’s half a dozen a *day*, then maybe nightly refreshes are a better fit. Personally, I schedule a nightly refresh at 2AM in my crontab file with the following entry:

```
# Refresh the main Giocosos database nightly at 2AM
# -----
0 2 * * * /usr/bin/giocosos.sh --refreshdb --dbname=main --musicdir=/sourcedata/music/classical
```

Remember to use the full path to the Giocosos script (i.e., /usr/bin/giocosos.sh) in crontab, rather than relying on just ‘giocosos’, as that requires a PATH variable to work, and PATHs are seldom set correctly in the context of a crontab job.

9.3 Table Structures and Methodology

In normal use, you do not need to know, nor care, about the inner structures of the Giocosos database - though if you want to construct your own SQL reports and analyses of your music collection, understanding its tables and their contents may be helpful. If you are not skilled in the ways of SQL, however, you can ignore the next few pages: skip to ‘Accessing the Database’ further on!

The Giocosos database, when created, contains just five tables, as follows:

- **musicdir:** a one-row table that contains a record of the ‘source’ folder within which all music files can be found, and the highest play number ever exported in a differential report
- **plays:** every time a composition is played to completion, a record is inserted into the PLAYS table to record the date, time, composer, composition, genre and duration
- **recordings:** populated after the music collection is scanned as part of a createdb or a refreshdb operation. One row represents an entire composition, with details of composer, genre and total duration.
- **sqlite_sequence:** an internal table that records the unique ID numbers given out to recordings, and plays when those tables are populated
- **tracks:** a table that stores one row per FLAC file discovered during a scan of a music folder during createdb or refreshdb operations. A four movement symphony will be represented by four rows in the tracks table, therefore.

Note that a scan of a music collection only directly populates the TRACKS table, where one row represents one physical FLAC discovered during the scanning process.

The RECORDINGS table is populated by aggregating the TRACKS records, grouping by unique composition name. A row in the RECORDINGS table will thus represent a four-movement symphony by a single row, with its aggregate duration being the sum of the durations of each of the files that belong to that symphony. The metadata tags (including album art) associated with the first file is assumed to apply in the aggregate to the ‘whole composition’. Thus, if it happened that you’d tagged track 01 as being by ‘Beethoven L’ and the other three tracks as being by ‘Ludwig Beethoven’, the entire composition would be attributed in RECORDINGS to ‘Beethoven L’.

When making a choice of something to play, Giocosio selects from the RECORDINGS table, never the TRACKS. It doesn't even display per-track data, so the contents of the TRACKS table is really irrelevant to Giocosio's operation immediately once a scan of the music collection has been completed.

Put another way, if you run Giocosio with `--composer=xxxx` or `--genre=yyyy` parameters, the table being queried to find music by a particular composer or belonging to a particular genre is always the RECORDINGS table.

When making a random selection of something to play, Giocosio makes two queries of the RECORDINGS table, in fact. First, it "selects distinct composer from recordings" so that it essentially produces a 'virtual table' of composers, where each composer is represented by a single row. That means every composer has an equal a chance as any other to be selected. It therefore makes no difference if you own 4000 Bach recordings and only 2 by James Macmillan: when you select distinct composer, you'll end up with 1 row each for both Bach and Macmillan and (crucially) each is as likely as the other to be selected.

Once a composer has been randomly selected, Giocosio then makes a second query of the RECORDINGS table to find any music by that composer, and chooses one row from available candidates. That then provides it with the 'DIRNAME' (i.e., the directory name or full path) to where the music associated with that composition can be found on hard disk. It then proceeds to play the physical files found in the location indicated.

The TRACKS table has no indexes, since it is not queried after it has been used to populate the RECORDINGS table. All other tables have a primary key on an auto-incrementing ID column (that is, the schema uses synthetic primary keys throughout). Importantly, this ID is **not invariant**. At each database re-scan, the TRACKS and RECORDINGS tables are completely cleared and re-populated afresh. This means a recording may acquire a different ID from the one it had before the database refresh. The ID is therefore not to be used to indicate anything of permanent significance.

Apart from the ID primary keys, the RECORDINGS table has secondary indexes created on a combination of composer+duration columns; and on the dirname column.

Note that the PLAYS table is **never cleared**. No matter how often you re-scan your music collection, the PLAYS table is left intact, so that it represents the accumulation of all plays made over time. Each new play is auto-assigned a synthetic primary key ID number: this remains the ID of that play for the life of the database, but has no relation to any similar-looking ID stored in any other table. (It does not represent a 'composition name' across all tables, for example).

9.4 Schema Description

The schema design uses no foreign key relationships: each table is entirely standalone and independent of the others (though the content of RECORDINGS is directly derived from TRACKS, of course).

The following screenshots attempt to show the entire Giocosio schema (the database name will depend on what parameter is supplied to the `--createdb=xxxx` parameter).

MUSICDIR

musicdir	CREATE TABLE musicdir (id integer not null check(id=1),musicdir text, maxplayexport numeric default 0, primary key(id autoincrement))	
id	integer	"id" integer NOT NULL CHECK("id" = 1)
musicdir	text	"musicdir" text
maxplayexport	numeric	"maxplayexport" numeric DEFAULT 0

PLAYS

plays	CREATE TABLE plays (id integer not null, PLAYDATE text, DIRNAME text, COMPOSER text, COMPOSITION text, GENRE text, PERFORMER text, DURATION numeric, primary key (id autoincrement))	
id	integer	"id" integer NOT NULL
PLAYDATE	text	"PLAYDATE" text
DIRNAME	text	"DIRNAME" text
COMPOSER	text	"COMPOSER" text
COMPOSITION	text	"COMPOSITION" text
GENRE	text	"GENRE" text
PERFORMER	text	"PERFORMER" text
DURATION	numeric	"DURATION" numeric

RECORDINGS

recordings	CREATE TABLE recordings (id integer not null, DIRNAME text, COMPOSER text, COMPOSITION text, GENRE text, COMMENT text, PERFORMER text, DURATION numeric, primary key (id autoincrement))	
id	integer	"id" integer NOT NULL
DIRNAME	text	"DIRNAME" text
COMPOSER	text	"COMPOSER" text
COMPOSITION	text	"COMPOSITION" text
GENRE	text	"GENRE" text
COMMENT	text	"COMMENT" text
PERFORMER	text	"PERFORMER" text
DURATION	numeric	"DURATION" numeric

TRACKS

tracks	CREATE TABLE tracks (DIRNAME text, COMPOSER text, COMPOSITION text, GENRE text, COMMENT text, PERFORMER text, TRACKNUMBER text, TITLE text, DURATION numeric)	
DIRNAME	text	"DIRNAME" text
COMPOSER	text	"COMPOSER" text
COMPOSITION	text	"COMPOSITION" text
GENRE	text	"GENRE" text
COMMENT	text	"COMMENT" text
PERFORMER	text	"PERFORMER" text
TRACKNUMBER	text	"TRACKNUMBER" text
TITLE	text	"TITLE" text
DURATION	numeric	"DURATION" numeric

INDEXES

Indices (3)		
plays_dirname_idx	CREATE INDEX plays_dirname_idx on plays (dirname)	
dirname	"dirname"	
rec_composer_duration_idx	CREATE INDEX rec_composer_duration_idx on recordings (composer, duration)	
composer	"composer"	
duration	"duration"	
rec_dirname_idx	CREATE INDEX rec_dirname_idx on recordings (dirname)	
dirname	"dirname"	

A textual representation of the schema is available from within the sqlite command line shell, which can be accessed from a terminal session with the following commands (which assume the database is named “main”, and thus exists as a file called main.db within the standard Giocosio folder:

```
cd $HOME/local/share/giocoso
sqlite3
.open main.db
.tables
.indexes
.schema
.quit
```

(It is the `.schema` command that will display the SQL code needed to re-create the schema completely)

9.5 Accessing the Database

The PLAYS table can be exported (to other databases, or to spreadsheets and so on) using the **--report** runtime parameter.

The RECORDINGS table can be queried in summary form using the **--stats** runtime parameter.

Since the Giocoso database is a standard Sqlite3 database, it can be queried at any time using standard (and free/open source) tools such as the [DB Browser for Sqlite](#).

The standard Sqlite3 command line shell can also be used to query the Giocoso database directly, independently of Giocoso itself.

Do please note that though the database is configured to run in ‘Write Ahead Logging (WAL) mode’, such that a reader shouldn’t block another reader, nor a writer a reader, if you are accessing the database at the same time as Giocoso is trying to record a fresh ‘play’ in it, unexpected results may occur. It is also generally not a good idea to perform your own inserts, updates or deletes on any part of the Giocoso database using external tools: corruption to the database (and hence the loss of the PLAYS history) may occur if you try.

9.6 Backing up the Database

It is important that you back up your Giocoso database: the PLAYS history cannot be re-generated should anything befall the database (such as internal corruption). The database is physically stored as a single file, called ‘*database-name.db*’ in the **\$HOME/.local/share/giocoso** folder. The fullstop (period) in front of the ‘./local’ bit of that path means the folder is *hidden*: you may need to turn on ‘show hidden files’ in your browser before you can navigate to it. It is probably a good idea to make sure backups are done to a non-hidden folder on your backup device, so that you can find it again easily if you need to! A command such as:

```
cp $HOME/.local/share/giocoso/* /backup/
```

...should be sufficient to capture the entire Giocoso database components **and** make them easily visible in the backup directory.

Bear in mind that you’re backing up the entire database really only because the PLAYS table cannot be reconstructed. If you play music frequently, therefore, the frequency of your backups needs to be higher than if you only play music occasionally (and thus only update PLAYS occasionally). Personally, I do a nightly backup of my Giocoso databases with the following crontab entry:

```
# Nightly backup of Giocosos at 4AM
# -----
0 4 * * * /usr/bin/rsync -avh --delete $HOME/.local/share/giocosos/
/home/hjr/Documents/Backups/Giocosos_Backups/
```

The last line is actually one continuous line, but wraps here for formatting reasons. Rsync is used rather than a straight copy so as to be sure that the backup folder contains only one copy of the database, overwritten nightly. I should perhaps mention a backup from one local drive to the same local drive wouldn't really be a lot of use in the event of that hard drive failing! Fortunately, in my case, the initial Giocosos backup destination is itself backed up multiple times to multiple remote devices frequently, so that's not a consideration here!

9.7 Recovering the PLAYS table

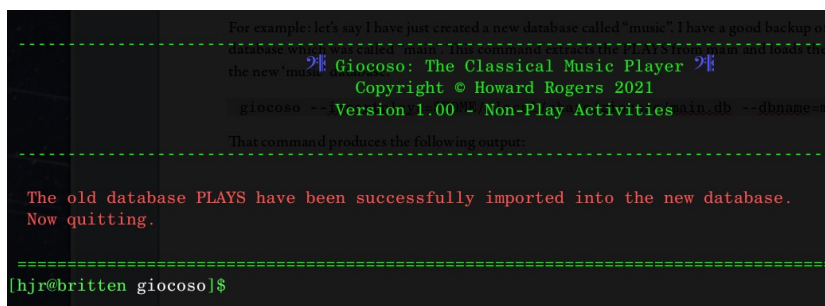
The RECORDINGS and TRACKS tables are, functionally, replaceable: if anything happens to them, they can simply be re-built or re-populated by re-scanning your music collection. But the PLAYS table is much more of a problem: lose that data, and it's gone for good.

Fortunately, Giocosos allows you to re-load plays into a database from which the originals have, for whatever reason, been lost, using the **--importplays=xxxx** run-time parameter (where 'xxxx' is the path and filename to a backup copy of your database, taken before the data loss occurred).

For example: let's say I have just created a new and mostly database called "music". I have a good backup of my old database which was called "main" -and thus the file containing it was called 'main.db'. This command extracts the PLAYS from the old 'main' and loads them into the new 'music' database:

```
giocosos --importplays=$HOME/.local/share/giocosos/main.db --dbname=music
```

That command produces the following output:



```

For example: let's say I have just created a new database called "music". I have a good backup of
database which was called "main". This command extracts the PLAYS from "main" and loads them
the new "music" database.
  2E Giocosos: The Classical Music Player 2E
      Copyright © Howard Rogers 2021
      Version 1.00
giocosos -- Version 1.00 Non-Play Activities main.db --dbname=music

That command produces the following output:

The old database PLAYS have been successfully imported into the new database.
Now quitting.
=====
[hjr@britten giocosos]$
```

Continue using the new database to play more music and the new play records will be added to those rescued from the old database: the continuity of your plays will not be interrupted (much!). The plays are not *deleted* from the old database file, by the way: they are merely *copied* to the new one.

Note that this is also the way you can transition from using the earlier AMP program (Absolutely Batching Music Player) to Giocosos without losing the PLAYS already recorded in AMP: simply point the --importplays parameter at the old AMP database file and its plays will come across to the new Giocosos one without much effort.

10.0 Run-Time Parameters

Running Giocososo in Random Play Mode (i.e., with a database) allows you to get quite creative in the things you can ask Giocososo to play for you: something that's a ballet conducted by Solti that takes less than 50 minutes to play? Not a problem. Something that's not an opera, but was written by Wagner? Check. Something that's an opera, but doesn't have Giuseppe di Stefano singing and which lasts no more than 2 hours? Fine.

You get the idea, I hope: you can pile on the filters and selection criteria so that Giocososo starts to play quite focussed or tailored types of music, depending on your mood for the day!

Or you can just forget all that cleverness and ask Giocososo to play 'something, anything'.

The way you decide to run Giocososo is really up to you -but getting subtle about it requires that you master the use of the various 'run-time parameters' (sometimes called 'switches') that are available to control its behaviour. These all take the form of double-dash keywords appended to the basic 'giocososo' command, and (depending on the keyword) assigning a value to the keyword.



There are currently a whopping 36 run-time parameters that you can supply when launching Giocososo, using a double-dash prefix.

That sounds like a lot, but they can be thought of as existing in four distinct 'groups', some of which are going to be rarely used (or even used only once).

Just 15 parameters control the sorts of things you are going to want to specify quite regularly. These are detailed in Section 10.3 below, as **Selective Parameters**.

A handful of parameters control the look-and-feel of the program, which you may decide you never need to alter anyway: see Section 10.2 below: **Appearance/Behaviour Parameters**.

A further half-dozen or so parameters are quite important to use semi-frequently. They control things like checking and getting Giocososo updates, or creating and refreshing a database of music. These are listed in Section 10.1 below: **Initialization Parameters**.

Finally, there are a bunch of parameters that allow you to report on the state of your music database. You may never need or want to know how much of it you've played, or what the continuous play-time of the entire collection is... but if you've ever wondered, Section 10.4 **Reporting Parameters** will be useful.

In the following sections, I'll simply list the parameters in a table, with a very summarised description of their purpose. I'll then explain them in much more detail after the table listing.

Please note that there's one parameter I do not discuss here, but in **Appendix C**, for reasons that appendix hopefully makes clear.

10.1 Initialization Parameters

These parameters are usually only used once (or infrequently), as part of post-installation configuration or tailoring.

<code>--createdb=xxxx</code>	Instruction to create a database, with <code>xxxx</code> as the database name.	Requires <code>--musicdir</code> parameter (otherwise, it doesn't know where to scan to populate the new database).
<code>--refreshdb</code>	Instruction to rescan a music collection and re-populate whatever database is specified with the accompanying <code>--dbname</code> parameter	Needs <code>--dbname</code> . Also needs <code>--musicdir</code> , otherwise it won't know what files to re-scan for changes.
<code>--musicdir=xxxx</code>	Specifies the physical location of a folder tree structure containing music (FLAC) files. Needs <code>xxxx</code> to represent a full path to the root of the collection.	Needs to be present when creating a new database and when refreshing the database. Not needed in regular play activities.
<code>--importplays=xxxx</code>	Instruction to extract the PLAYS table records from another database and copy them into the database specified with the <code>--dbname</code> parameter. Basically allows old PLAYS to be moved between databases (including old AMP ones). Allows recovery of PLAYS from backup.	Needs the full path and filename to the original AMP database, which is usually found within the <code>\$HOME/.local/share</code> folder. Also needs the <code>--dbname</code> parameter so we know what database to load the data into.
<code>--integrate</code>	Instruction to create appropriate files that enables right-click play of folders in Giocosio from your distro's GUI file manager	Only needs to be run once. See Section 7.0 for details of what happens and the limitations on various desktops/distros.
<code>--scrobble-config</code>	Instruction to initiate the authorisation mechanism that lets Giocosio scrobble its plays to Last.fm	Only needs to be run once. Needs a valid last.fm account before you start. See Section 8.0 .
<code>--checkver</code>	Instruction to check to see if a newer version of Giocosio has been released and to fetch and install it if one is found.	Will prompt for sudo privileges if a new version is found (as sudo is required to copy relevant files to privileged directories).
<code>--licence</code> <code>--license</code>	Instruction to display the GPL v2 text under which Giocosio is licensed.	US and British spelling variants both work the same

These ‘administrative parameters’ tend to be used only once -or, at least, very infrequently. Usually, you’d use them immediately after a fresh installation of Giocoso, because they setup and configure things which enable much of Giocoso’s functionality. In the sections that follow, I remove the leading ‘--’ before the parameter name for aesthetic reasons: the double-dash is needed when actually using the parameters, though!

10.1.1 Createdb

The **--createdb** parameter triggers the creation of a Sqlite3 database, with a name that matches the parameter value supplied. If you said **--createdb=mymusic**, for example, then the database ‘mymusic’ would be created and a physical file, stored in **\$HOME/.local/share/giocoso**, named **mymusic.db** would be created.

Since the database name becomes a physical file name, it’s a good idea to make your database name a simple, short, one-word value without spaces. Good names might be, for example, ‘music’ or ‘main’ or ‘overflow’. A bad name might be something like ‘My main database’, because all those spaces might cause trouble further down the line. If you insist on using spaces within a database name, you’ll have to wrap the entire parameter within double quotation marks, so you’d have to end up saying something like ‘--createdb=”My Main Music Database”’.

I would recommend not using capital letters either: Linux filesystems are case-sensitive, so if you mix cases in the database names, you’ll always have to remember the precise combination of upper- and lower-case letters for all future uses. Sticking to one-word, lower-case database names makes life easier all round!

The **--createdb** parameter always has to be used in combination with the **--musicdir** parameter. That’s because immediately after creating the database file, Giocoso scans the musicdir provided to populate it with data. If it doesn’t know where to scan, because you didn’t mention a musicdir, database creation will fail, with an appropriate error message displayed.

The full database creation command would therefore be something like:

```
giocoso --createdb=music --musicdir=/music/classical
```

This will create a database file called **\$HOME/.local/share/giocoso/music.db** and then scan the entire contents of the **/music/classical** folder to populate that new database.

Note that you can create as many databases with Giocoso as you like. If you have classical, jazz and ‘not very good’ music in distinct folder hierarchies, for example, you can easily create three different databases called (perhaps) ‘classical’, ‘jazz’ and ‘overflow’. You can later choose which database to source a ‘play’ from by running Giocoso with a particular value for the **--dbname** parameter (as we’ll see shortly).

10.1.2. Refreshdb

A static music collection may be scanned just the once, as part of the createdb process, and need never be scanned again: it’s static, after all, and thus not changing, so the snapshot of it taken at database creation time is a true picture of that collection.

If, however, you are still acquiring new music to add to your collection, this one-off ‘snapshot scan’ of a folder structure at database creation time will not meet your needs: if the collection is changing, the database contents need to be periodically refreshed to make it keep up with those changes, which is why you’d use the `--refresh` database parameter in a command such as:

```
giocosos --refreshdb=music --musicdir=/music/classical
```

This triggers Giocosos to take a fresh look at the contents of the folders contained within the folder again specified by the `--musicdir` parameter value, and to re-load the database with details of whatever FLAC files it now finds there. Depending on how often you are adding to your music collection, you may need to schedule a database refresh using *cron* maybe nightly, maybe weekly -or, at least, at whatever frequency you think is required to keep the Giocosos database useful. Giocosos cannot randomly select to play recordings of whose existence it is unaware, anyway.

My own crontab file has this entry, for example:

```
# Refresh the main Giocosos database nightly at 2AM
# -----
0 2 * * * /usr/bin/giocosos.sh --refreshdb --dbname=main --musicdir=/music/classical
```

(Remember that in crontab, one tends to specify the full path to executables, so here it’s `/usr/bin/giocosos.sh`, not just ‘giocosos’). You can see that I refresh my database nightly at 2AM, when no-one’s around to see it happen!

Note that a database refresh *clears the existing content of the database tables and re-populates them from scratch*. Giocosos does not, in other words, only ‘add in the new stuff discovered’. It adds in everything, starting from a clean slate. (Play history is always preserved, however!). This means that a refresh database can take quite a long time to complete, since it’s scanning the whole music collection again, not just the new additions to it. Practically, it also means that recordings acquire different ID numbers in the database at different times. A recording of Britten’s *Peter Grimes* might be assigned ID 1583 one day and become recording ID 1586 the next, because 2 new acquisitions in the ‘A’ composers have bumped it down the list. Never rely on recording IDs being invariant between refreshes, basically.

10.1.3 Musicdir

As already mentioned, the `--musicdir` parameter takes a value of the full path to the root of a folder tree containing a set of FLAC files. It is required for database creation and refreshes.

In case it’s not obvious, you can only specify *one* music folder when creating or refreshing a database, so the path supplied must be ‘high up enough’ to allow the collection scan to capture all appropriate music in one pass.

If you had a folder structure like this, for example:

```
/Benjamin Britten
/Classical Music/Beethoven
/Gustav Mahler
```

...it would be difficult to have all three composers included in one database unless you said your musicdir was '/' (the root folder)... and a scan of your system's root folder is likely to be long and painful!

Better would be something like this:

```
/music/Benjamin Britten  
/music/Classical Music/Beethoven  
/music/Gustav Mahler
```

...for now you can scan '/music' and get all three composers collected in one hit. It doesn't matter for these purposes that Beethoven is one folder further 'down' than the other two: Giocosos will find music within a top-level folder no matter how far down the 'tree' it might be. There are aesthetic reasons for wanting a consistent folder hierarchy and structure, however, but core playback functionality is not affected by such deviations.

10.1.4 Importplays

Moving your play history between databases is probably not something you'll do very often, but the **--importplays** parameter allows you to do it. The parameter takes a full path and filename argument, pointing to the Sqlite database containing the *source* play history. You must also supply a **--dbname** parameter so we know what database is to be the *destination* of the import process.

Source databases must be either Giocosos databases or AMP ones (AMP is the Absolutely Baching Music Player, which was the prototype for Giocosos: my own AMP play history is now over 6 months long and I didn't want to lose it when transitioning to Giocosos -so, being able to pull it out of AMP's database and into Giocosos's was very important to me).

Since the source database can be another Giocosos database, this is the parameter you'll want to use to perform restores of a play history from a backup of your main Giocosos database. The command would be something like:

```
giocosos --importplays=/backups/mymusic.db --dbname=mymusic
```

There is nothing to stop you running this sort of import process multiple times -but if you do, the freshly imported records merely duplicate the records you previously imported. Each new import will double the number of plays being stored in Giocosos's PLAYS table. To clear them, you'd have to physically delete the Giocosos database, create a new one and do a single, fresh import.

10.1.5 Scrobble-config

Giocosos is a 'scrobbling FLAC player', meaning that it can transmit a record of what it plays to Last.fm, where a web-accessible play history can be built up over time. For it to be able to scrobble regularly, however, Giocosos needs to be put through a one-off authorisation process. The **--scrobble-config** parameter is used to run that authorisation process. See **Section 8.1** for details. Once you have run the scrobble-config process, you need never run it again.

10.1.6 Checkver

The **--checkver** parameter is something you will want to run infrequently, but regularly (say, once a week or once a month). The command:

```
giocosos --checkver
```

...simply forces Giocosos to check the Internet to see if a more recent version of the program available. If there is, Giocosos will offer to fetch and install it for you. If there isn't, it will simply say 'you're already running the latest version'.

Should a more recent version be discovered, you will be asked whether you want it fetched and installed. If you agree to do so, you will be prompted to supply your sudo password (because installing Giocosos involves writing into the /usr/bin folder, which ordinary users cannot do, but which root can). The upgrade will always overwrite the *program* files involved, but will never alter in any way your music database. Giocosos upgrades are therefore quite safe (though you're still always recommended to take a backup of your database before doing one, just in case!)

It is important to perform this version check reasonably often, because new software releases fixing bugs and adding new functionality are being continuously made, with no fixed schedule.

10.1.7 License/Licence

The **--license** parameter can be spelled in either US or British variants, but no matter how you spell it, its presence causes the full text of the GPL v. 2 license to be downloaded and displayed within the **less** program (meaning that you can up- and down-arrow or PgUp/PgDn to scroll your way through it in either direction). Press 'q' to quit the display and exit back to the command line. The parameter has no real utility, other than to conveniently display the full text of the license under which Giocosos is made available. It's not an option you're going to use very often, I think!

10.2 Appearance/Behaviour Parameters

These parameters tend to affect the way the program looks or behaves, at a fairly high level. Many people will never need to use any of them at all... but if you *do* decide to use any of these parameters, you will probably end up using them every time you run Giocosos -perhaps via an alias (see [Section 11.5](#) for a discussion of the use of aliases to make running Giocosos easier).

--displaycolour= --displaycolor=	light, dark, neutral, standard, classic	Alters the colour display of the text. The default is 'standard'
--artsize=	small, medium, large, none	If album art is displayed, what size to display it at. Small=300x340. Medium=450x500. Large=900x960. None switches art display off. The default is 'medium'.
--captiontext=xxxx	Sets colour of caption text displayed with album art	Defaults to white. 681 alternatives are available, provided the supplied value matches one of the colours displayed with --listcolours.
--captionbackground=xxxx	Sets the background colour of the caption displayed with album art	Defaults to black. 681 alternatives are available, provided the supplied value matches one of the colours displayed with --listcolours.
--listcolours --listcolors	Lists valid colour values for use with caption options	Colours are listed inside the 'less' program, so it's scrollable in either direction. Press q to quit back to command line.
--scrobble	Instruction to scrobble	Submits a 'play' to last.fm on play-completion if supplied. The default is not to scrobble.
--device=	default or some other hardware value, system- dependent	The audio device to stream output to. The default is 'default'. Knowledge of specific hardware addressing required to supply any other valid value
--editor	Instruction to edit the excludes.txt. Present only in version 1.07 and above.	Opens excludes.txt in whichever text editor is set as the value of the EDITOR environment variable. The contents of the excludes.txt determines whether certain composers' music should be excluded from random selection. See Section 5.2 for details of the excludes mechanism.

10.2.1 Displaycolour/Displaycolor

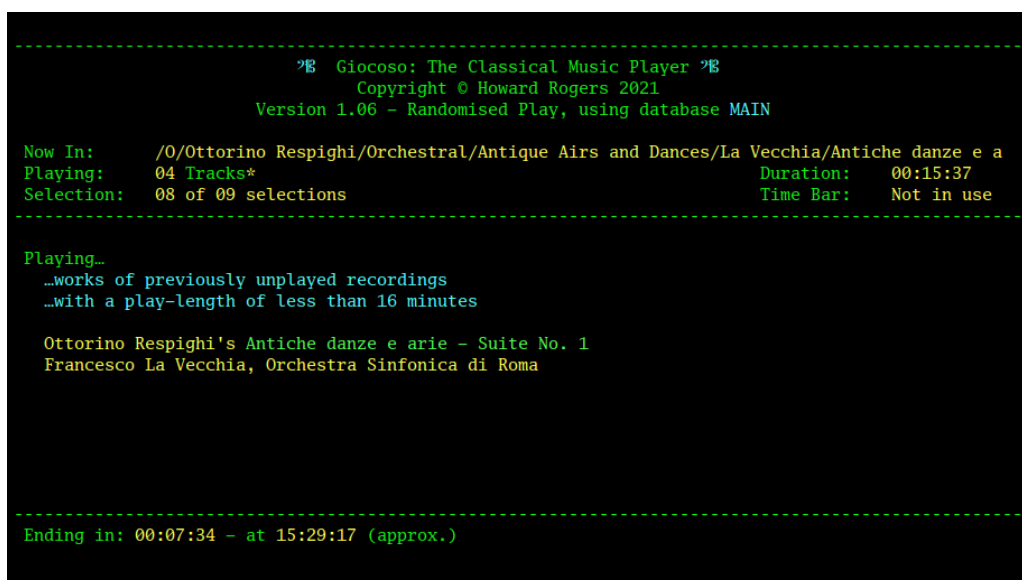
The `--displaycolor` parameter can be spelled in US or British variants, and takes one of four possible values, with 'standard' being the default. Any value passed that isn't 'light', 'dark' or 'neutral' is silently interpreted as 'standard'.

The standard colour scheme involves the use of cyan, green, red and yellow text: users with colour perception issues may prefer trying one of the other three values, which essentially turn the display into monochrome white ("light"), monochrome black ("dark") or monochrome yellow ("neutral").

Note that the various colour schemes might not always display correctly unless you *also* configure the appearance of your terminal session. If you ask Giocososo to display 'dark', for example, when your terminal profile already mandates a dark appearance, you'll be basically looking at black text on a black background!

Here are screenshots of the various colour options available to you via the use of this parameter.

First, standard:

A screenshot of a terminal window showing the Giocososo interface. The text is displayed in various colors: green for static program text, yellow for dynamic content, red for error messages, and cyan for user-dependent static text. The interface includes a header with the program name and version, a status bar with current track information, and a main display area showing the current track and its details.

```
-----
                                ⌘ Giocososo: The Classical Music Player ⌘
                                Copyright © Howard Rogers 2021
                                Version 1.06 - Randomised Play, using database MAIN
Now In:      /0/Ottorino Respighi/Orchestral/Antique Airs and Dances/La Vecchia/Antiche danze e a
Playing:     04 Tracks*
Selection:   08 of 09 selections
Duration:    00:15:37
Time Bar:    Not in use
-----

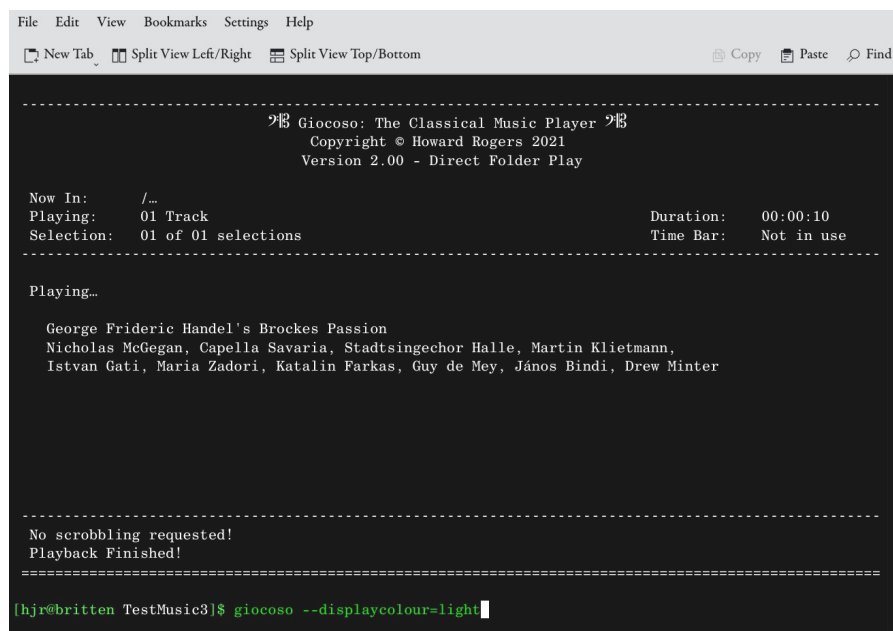
Playing...
...works of previously unplayed recordings
...with a play-length of less than 16 minutes

Ottorino Respighi's Antiche danze e arie - Suite No. 1
Francesco La Vecchia, Orchestra Sinfonica di Roma

-----
Ending in: 00:07:34 - at 15:29:17 (approx.)
-----
```

Broadly speaking, the standard display uses Green to display static program text; Yellow to display dynamic text whose content will depend (maybe indirectly) on something the user supplied; Red to display error messages or alert condition warnings; and Cyan (light blue) to display static text that depends on user input (such as the name of the database in use).

Now, Light:



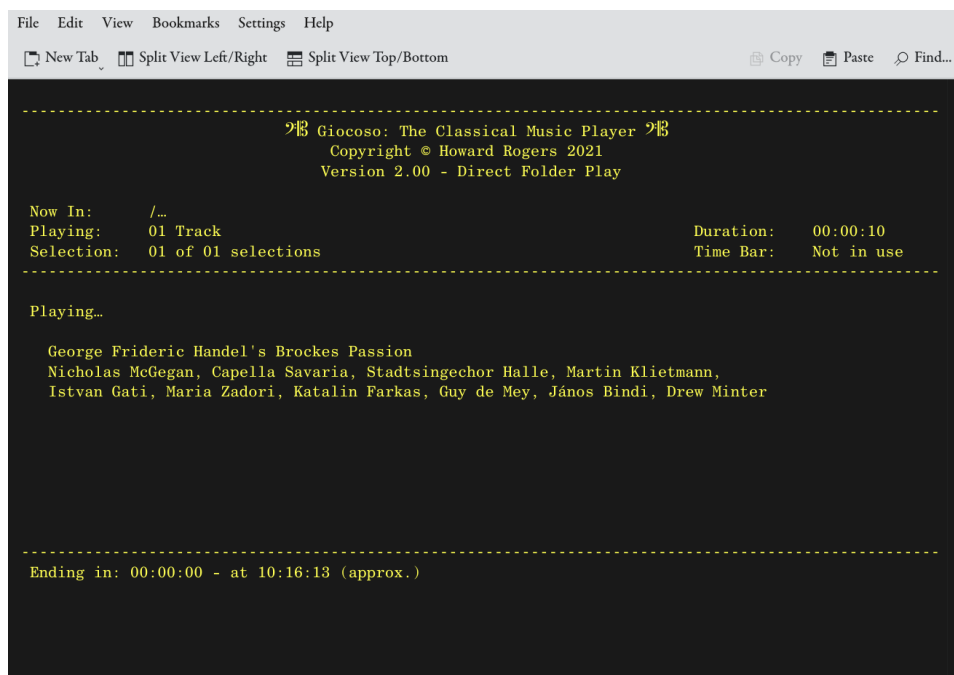
Everything becomes white monochrome text, basically, so that there's no difference between program text, user information and error messages.

Now Dark:



Here, I've altered the Konsole theme to be 'Solarised Light'... and now you can see that the dark Giocosso theme results in monochrome black text on a now-white background.

Finally, here's Neutral:



As you can see, this is essentially monochrome yellow -the idea being that yellow-on-black is less tiring on the eyes than plain black-on-white or white-on-black.

In earlier versions of Giocosso, the default colour scheme used a dark blue (sort of a Royal Blue) to display static text whose value depends on user selections or input. The dark blue was so dark, however, that reading it could sometimes be a problem. From version 1.05 onwards, the dark blue became the lighter cyan colour you see in the screenshots for the 'standard' (and default) colour scheme. Should you still wish to use the darker blue, however, you can do so by specifying the **--classic** colour scheme. Apart from the dark blue/cyan switch, the classic colour scheme is identical to the current 'standard' one.

10.2.2 Artsize

Giocosso is able to display embedded album art (that is, artwork which has been added into the PICTURE tag of a FLAC file). It cannot display artwork which is external to the music file (so, it does not process folder.jpg files, for example).

By default, if album art is detected, it will be displayed at 450x500 pixels, usually in the lower-middle of your screen (though the precise location of the display window can be a little unpredictable, depending on what else is running on your PC at the time a new folder of music begins to be played!) This corresponds to setting the **--artsize** parameter to a value of medium. The parameter also accepts values of **small** (300x340) or **large** (900x960). Any other value you supply is interpreted as medium.

Whatever the size of the album art display you choose, an additional 40, 50 or 60 pixels is bolted onto the bottom of the album art proper, with a black background (by default) and the first 80 characters resulting from the combination of the last name of the ARTIST tag plus the ALBUM tag, displayed in white text (by default). Just by glancing at the album art display, therefore, you should be able to see exactly what piece is playing, and who the composer is... and if the album art is helpful rather than just 'artistic', you'll probably also know the performers, conductor and so on.

The album art window, once displayed, can be moved around by mouse clicking-and-dragging as you like; you can also shut the artwork window down at any time, though you will not be able to re-open it afterwards. Closing the artwork display won't stop music playback, but terminating music playback (by pressing Ctrl+C in the Giocosso main window) will immediately close the album artwork window.

The parameter can also be set to a value of **none**. This prevents any artwork at all being displayed.

10.2.3 Captiontext

As mentioned above, if Giocosso displays album art, it will grab the last of the names listed in the ARTIST tag, join that to the ALBUM tag of the first FLAC file found to play, and then grab the first 80 characters from the combination. The result will then display below that art work -and it will default to displaying that text in white lettering. If you would prefer a different colour, then you can tell Giocosso what colour to use with the **--captiontext** parameter.

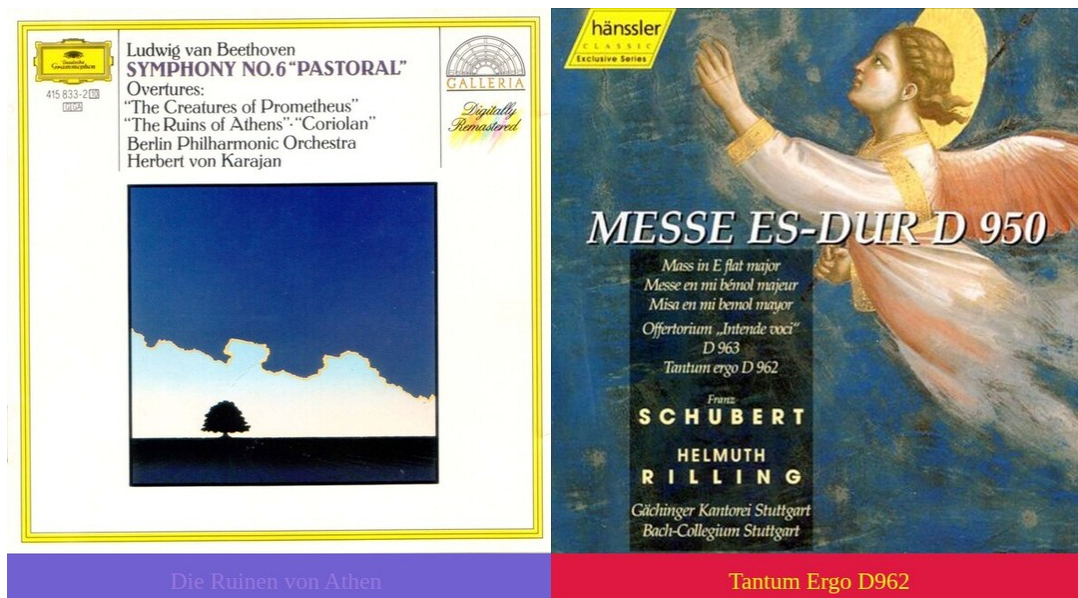
Valid values for this parameter can be listed with the **--listcolours** parameter (see below, **Section 10.2.5**). The parameter is, however, case insensitive, so whether you say **--captiontext=cyan** or **--captiontext=CyAn**, you'll get cyan-coloured text. If you supply an invalid colour (and there are 683 of them to choose from!), then the parameter defaults back to plain white.

10.2.4 Captionbackground

Just as you can alter the colour of the album art caption *text* (see **Section 10.2.3** above), so you can alter the colour of the caption background with the **--captionbackground** parameter. It defaults to being black, but this parameter allows you to specify an alternative colour, providing that the parameter value is a colour that is amongst those listed when you use the **--listcolours** parameter (see below, **Section 10.2.5**).

The parameter is case-insensitive, so if you say the background should be DarkGreen or darkgreen or DaRkGrEeN, you'll get a dark green background just fine. Any invalid colour names, however, are silently defaulted back to plain black.

To give you an idea of what's possible, here are two displays of album art+caption, one using **--captiontext=mediumpurple** and **--captionbackground=slateblue**; the other used **--captiontext=yellow** and **--captionbackground=crimson**. I don't say these are particularly *good* colour combinations! But they at least give you an idea of the range of what's possible and the sort of visual consequence of your choice(s) you can expect:



Note that if Giocosos spots you've specified the same colour for the text and the background, it will silently give you white text on a black background: it's not going to display just a coloured bar with no apparent text in it! Do note, however, that some distros (notably Ubuntu and its derivatives) default to not permitting text to be converted to images -and if you fall foul of that restriction, you definitely *will* end up with a coloured bar with no apparent text in it.

If that happens to you, you will need to relax the security provisions of the `/etc/imagemagic-6/policy.xml` file (though your particular distro may introduce upper case characters into the name, and the version number (6) might vary, too).

The specifics of this (should you need it) are discussed on my website at the following address:
<https://absolutelybaching.com/giocosos-on-kubuntu/>

10.2.5 Listcolours

To use the `--captiontext` and `--captionbackground` parameters (see above, **Sections 10.2.3 and 10.2.4**) successfully, you must supply valid colour values. There are 683 distinct colour 'names' to choose from: the `--listcolours` parameter (also `--listcolors` if you prefer that spelling!) will display a list of all valid colour names inside the 'less' text display program -meaning that you can arrow up and down the list in either direction one line at a time, or PgUp and PgDn through it rather more quickly.

When you are ready to quit the display of colours, just tap the letter 'q'. You will be returned to the command prompt from which you originally launched Giocosos.

10.2.6 Scrobble

The `--scrobble` parameter controls whether or not Giocosos should transmit details of what has just been played to the Last.fm website. Doing so builds up a web-accessible history of what music you listen to over time. This can be quite helpful in allowing you to spot listening trends (and maybe prompt you to modify your listening habits in various ways). It is strictly optional to do, though, and

Giocosos will **not** scrobble your listening habits unless you explicitly request it to do so by including the `--scrobble` parameter on the command line as you launch Giocosos.

The subtleties of scrobbling are more fully described under **Section 8** of this document. Be aware that even if you have configured to scrobble (that is, authorised Giocosos as a program that is allowed to post to your Last.fm account), it will not do so unless the `--scrobble` parameter is present on the command line as you launch Giocosos. Giocosos's default behaviour is **not** to scrobble, in other words, even though all necessary prerequisites may allow it to do so.

10.2.7 Device

The `--device` parameter is used to direct Giocosos's audio output to a specific hardware device, provided the device name is known. If not specified, the default device is simply 'default' -and this will work for most Linux environments I know of. If, however, you have external DACs or other complex audio hardware configurations, you may need to control more precisely where Giocosos directs its output.

In my case, for example, I run Giocosos with the command:

```
giocosos --dbname=main --device=plughw:2
```

...meaning that I am directing the playback to 'plugin hardware device with address 2,0' (which happens to be my USB-attached external DAC). How did I know that 'plughw:2,0' was the correct device identifier to use?

Well, it isn't terribly easy, I'm afraid. In the first place, I issued the command:

```
cat /proc/asound/cards
```

...and this output the following:

```
0 [PCH          ]: HDA-Intel - HDA Intel PCH
                  HDA Intel PCH at 0xfb320000 irq 54
1 [Nvidia       ]: HDA-Intel - HDA NVidia
                  HDA NVidia at 0xfb080000 irq 55
2 [E30          ]: USB-Audio - E30
                  Topping E30 at usb-0000:06:00.0-1.2, high speed
3 [C920         ]: USB-Audio - HD Pro Webcam C920
                  HD Pro Webcam C920 at usb-0000:05:00.0-3, high speed
4 [webcam       ]: USB-Audio - Full HD webcam
                  SunplusIT Inc Full HD webcam at usb-0000:06:00.0-1.4.4, high speed
```

That at least told me that if I wanted to direct sound output to my Topping E30 USB-attached external DAC, I'd have to direct output to hardware device 2.

Generally, you'll use a **plughw** type of address with Giocosos. Therefore, if you know your device is number 2, it's a fairly good bet that the correct value for the `--device` parameter is going to be **plughw:2**. If you have more than one device on an output, you'll need to add a '1' or a '2' to distinguish them, but that's not the case in the above output for me -hence, **plughw:2** is the correct value for me and my hardware setup. But note that device IDs can change as you plug things in and out over time! So don't get attached to any specific device ID, but get comfortable with the method of determining what the correct ID to use should be.

The default value for this parameter is usually fine. You only need to mess around with `--device` if you have a complex audio setup and/or Giocososo seems not to be handling your configuration correctly.

An environment variable, called `GIO_DEVICE=` can be set (in `.bashrc`, or any other environment-setting file), to an appropriate value so that you don't have to keep remembering to supply the parameter on the command line every time you want to launch Giocososo. That is, if I were to add the line:

```
GIO_DEVICE="plughw:2,0"
```

...to my `.bashrc` (and log out and log back in so that the new value is picked up), I'd be able to launch Giocososo with the plain command:

```
giocososo --dbname=main
```

....and my audio output would work fine, via my DAC, even though `--device` has not been specified at run time. If the environment variable is set to X, and a run-time `--device` parameter has been set to Y, it is the run-time parameter Y that takes precedence: it overrides the environment parameter for the duration of that run.

10.2.7 Editor

Giocososo in random-play mode will ordinarily select a composer at random and then randomly select a composition by that composer for play. If, however, a text file called **excludes.txt** is present in the `$HOME/.local/share/giocososo` folder, that file will be read before making a random selection of a composer. If a composer's name is found in that file (one name per line), then that exact spelling of the composer's name is completely excluded from further random selection. The exclusion is automatic and permanent, until such time as the relevant line in the `excludes.txt` is removed.

As a text file, the `excludes.txt` can be edited by any text editor with which you are familiar. Gedit, Kate, Notepadqq... you name it, any one of them can do the job.

However, starting in version 1.07 and up, Giocososo can be run with the `--editor` run-time parameter and the `excludes.txt` can then be edited directly in whatever text editor the `$EDITOR` environment variable is set to. On many distros, by default, that will mean that it's opened in `vi`; in most others, it will default to being edited in `nano`. Whatever the `EDITOR` variable is set to can, of course, always be changed (usually by setting it in the `$HOME/.bashrc` file).

Once you have edited the document in whatever editor is selected for the job, you can save and close the file, and Giocososo will itself quit, without attempting to play music.

The `--editor` run-time parameter is therefore merely a convenience: it means that editing a file Giocososo uses to determine what music to play randomly can be done from the same command line as you use to run Giocososo as a music player. If you prefer to open the `excludes.txt` as a separate exercise entirely, however, that older approach continues to work just fine, too.

10.3 Selective Parameters

This set of parameters is the largest -and the ones which you'll most commonly use every time you run Giocosio. They control what music Giocosio plays, and the way it plays it. In the following table, where I say a parameter takes a value of *xxxxx*, I mean it accepts textual entries; where I say it takes a value of *y*, I mean that it takes a numeric argument.

<code>--dbname=xxxxx</code>	The name of the database to use to make music selections.	Must already exist
<code>--selections=y</code>	The number of music plays to perform before stopping	Default is 1. Maximum is 99. After the number of selections has been played, Giocosio quits and needs to be relaunched from the command line for any further plays to take place.
<code>--pause=y</code>	Introduces a pause of <i>y</i> seconds between consecutive plays of music.	The default pause is 10 seconds. There is no limit to this parameter's upper value, but values less than zero are silently reverted to the default 10 seconds.
<code>--timebar=y</code>	The number of hours during which a composer who has had any piece of music played cannot have further music played.	Default is 6, maximum is 999 (which is about 42 days). The intention is to stop the same composer being played repeatedly within a short timeframe. 0 disables the feature entirely.
<code>--genre=xxxxx</code>	A filter to force the random selection of music that has been tagged (in the GENRE tag) as belonging to a specific genre.	The supplied parameter is wildcarded on the outside so a value of 'on' would match concerto and symphony , for example. Case insensitive.
<code>--composer=xxxxx</code>	A filter to force the random selection of music that has been tagged (in the ARTIST tag) as having been written by a specific composer.	The supplied parameter is wildcarded on the outside, so a value of 'ga' would match Elgar and Havergal Brian , for example. Case-insensitive.

--performer=xxxx	A filter to force the random selection of music that has been tagged (in the PERFORMER tag) as being performed by a particular ‘distinguishing artist’ (which will usually be the conductor, or the chamber ensemble, etc)	The supplied parameter is wildcarded on the outside, so a value of ‘ol’ would match Solti and George Malcolm, for example. Case-insensitive.
--comment=xxxx	A filter to force the random selection of music that has been tagged (in the COMMENT field) with a particular name (or part-name)	The supplied argument is wildcarded on the outside, so a value of ‘=cal’ would match Maria Callas and Pascal Devoyon Case-insensitive.
--composition=xxxx	A filter to force the random selection of music that has been tagged (in the ALBUM field) with a particular name (or part-name)	The supplied argument is wildcarded on the outside, so a value of ‘gri’ would match Peter Grimes and The Pilgrim’s Progress. Case-insensitive.
--recordnumber=y	A filter to force the play of a specific, uniquely-numbered recording.	The number supplied must match a specific recording, as stored in the ‘Recordings’ table in the database. Nothing will play if it doesn’t. See the reporting options to determine what number is assigned to which recording.
--unplayed	A filter to force the random selection of music written by a composer who has not previously had anything at all played.	You acquire a new recording by Anton Hickmauser: Hickmauser is an unplayed composer. This filter will force the selection of that new acquisition, because Hickmauser is an unplayed <i>composer</i> .
--unplayedworks	A filter to force the random selection of recordings which have not previously been played, even though their composer may well have had other recordings played before.	You buy a new recording of Beethoven’s 5 th . Beethoven has had loads of other music played. This filter will force the hitherto-unplayed recording to be considered for the next randomised selection.
--minduration=y	A filter to force the random selection of recordings which last for at least y number of minutes .	The default is 0, meaning ‘any recording at all matches’.

<code>--maxduration=y</code>	A filter to force the random selection of recordings which last for no more than <i>y</i> number of minutes .	The default is 525960 minutes, which is roughly equal to one calendar year and therefore practically means ‘all recordings match’.
<code>--negate</code>	Reverses the sense of other filters.	If present, it has the effect of inverting the sense of another filter that is present. Thus <code>--composer=britten --negate</code> means ‘play anything provided it is NOT written by Benjamin Britten’ and <code>--comment=callas --negate</code> means ‘play anything at all, so long as Maria Callas is not listed in the COMMENTS tag as one of the performers.’

10.3.1 Dbname

The `--dbname` parameter defines which database from which to select music to play. If not present, Giocosos will assume you want to use Direct Play Mode, and thus search for FLACs to play in the current folder. When present, Giocosos will search the named database’s Recordings table to determine what should be played next.

Thus, the simplest way to run Giocosos in this randomised mode is to issue the command:

```
giocosos --dbname=main
```

(Or whatever database name you actually created with the `--createdb` parameter, of course. ‘Main’ happens to be the name I use for my own database). The database name refers to a specific, physical file found in the `$HOME/.local/share/giocosos` folder. If you say `--dbname=main` then Giocosos will be looking for a database file called `$HOME/.local/share/giocosos/main.db`.

If Giocosos cannot find a file of the right name in that location, it will warn you of the fact and then quit.

If it *can* find a database file of the right name, it will immediately open it, make a random selection of a composer, and then a random selection of a recording by that composer.

This two-stage selection process is significant. By selecting from composers first, where every composer is represented by a single row in a table, every composer is given an equal chance of initial selection. That you have 4000 CDs by Bach and only 2 by Havergal Brian is irrelevant: both Bach and Brian are represented equally within the composer selection process and both are thus equally likely to be picked for play.

Only once a composer has been selected does Giocososo then concern itself with making a random selection of a recording written by that composer.

The net effect is that, over time, every composer's work gets played roughly as frequently as any other's. The distribution of your plays, per-composer, should be approximately equal, given a long enough play history.

Most of the other 'selective' parameters I'm about to discuss are intended to 'nudge' or 'influence' Giocososo away from this strictly-equal-chance way of selecting music: if you are desperate to listen to some Wagner or a song sung by Callas, then a purely randomised selection of music won't be what you're after! You'll want to be able to say 'play Wagner', 'play Owen Wingrave' or 'play something performed by Callas' and have Giocososo take account of your specific desires and requirements. See **Sections 10.3.5** and onwards for how you can influence this randomisation process by applying various selection filters to Giocososo at run-time.

10.3.2 Selections

The `--selections` parameter controls how many pieces of music Giocososo will play before quitting. Valid values are anything between 1 and 99, with a default value of 1 and any invalid values supplied being silently converted to 1.

The parameter can be used in both Direct Play and Random Play Modes -but, obviously, in Direct Play Mode there's only one folder being played, so a selections value higher than 1 just means that same piece of music gets played over and over again! In Random Play Mode, however, a different piece of music will be selected each time we finish one selection and begin another.

Having a large value for `--selections` in conjunction with the `--dbname` parameter therefore allows you to create a sort-of 'wall of ever-changing music', similar to what you might hear when tuning into your local classical music FM radio station (but without the annoying talking the presenters like to indulge in between pieces!)

Instead of inane patter from a presenter, each selection is separated from the next by a period of silence, the length of which is governed by the `--pause` parameter (see below).

10.3.3 Pause

The `--pause` parameter takes a numeric value from zero to anything at all: there's no upper limit, but negative values are not allowed and, if present, are converted to the default value of 10. The value is the number of **seconds between successive plays of music when `--selections` is greater than 1**.

A value of zero will mean that a series of (say) 4 selections will play one after the other, without significant pause between them, though there may still be a bit of silence as it takes Giocososo a finite amount of time to find another piece to play and start its playback. The default of 10 gives a reasonable period of 'reflective silence' after one has finished and before the next kicks in.

10.3.4 Timebar

The time bar is a feature by which Giocososo will not play anything *else* by a composer who has already had *something* played earlier in the day. Suppose, for example, that Giocososo randomly selects some

Beethoven to play at 9AM one day: how long should Giocososo wait before selecting something else by Beethoven (conceivably, even, the same piece) for playback?

The default answer to that question is ‘6 hours’ -which is to say that *the default time-bar in Giocososo is six hours long*.

The **--timebar** parameter, however, can be used to change the default to anything from 0 hours to 999 hours (which is about 42 days). If **--timebar=0**, then a play of composer X at 9AM doesn’t stop composer X being selected for another play at all. You might end up, by chance, with an entire morning of plays of Elgar, for example ☺ If the timebar is set to less than 0 or more than 999, the default value of 6 kicks in silently.

A setting of **--timebar=999** will mean, however, that if Elgar is picked for playing at 9AM today, 1st January, he won’t be a candidate for anything else to be played until (approximately) midnight on Friday, 12 February.

The point of the time-bar is to stop Giocososo over-playing some composers at the expense of others -and to stop you getting bored with whoever it randomly chooses for playback. Please note that Giocososo records the time of a ‘play’ at the point that it *finishes* playing. So if you start playing a piece of Elgar at 9am, but it doesn’t end until (say) 9.20am, then the timebar will only apply from 9:20am onwards.

If you need a longer ‘ban’ on a particular composer (or set of composers) then, rather than set an extra-long timebar parameter, you can instead configure an “exclude”. See **Section 5.2** for details on how to do this.

10.3.5 Genre

The **--genre** parameter allows you to influence the random selection process so that what Giocososo picks to play matches your supplied parameter value (or partly matches it). If you fancy listening to some opera, for example, then

```
giocososo --dbname=main --genre=opera
```

...will get the job done (obviously the database name needs to match your own choice of database name: ‘main’ happens to be mine).

The parameter value is matched to whatever you’ve tagged your music files with in the GENRE tag and the genres you chose to use when tagging your music obviously affects what sort of selection you can make with the parameter now. If you’ve tagged everything up as ‘classical’, then you can’t really use the **--genre** selection filter now to make one type of music get played in preference to any other type, for example. And if all your operas were tagged as ‘Music Drama’, then a search now for ‘opera’ isn’t going to match anything.

If, however, you’ve used the sorts of tags I recommend at <https://absolutelybaching.com/music-articles/a-list-of-classical-music-genres/>, then you’ll be able to use this parameter to get Giocososo to play some ballet, or some string quartets, or some film music. The specifics of what will be picked to be played if you say **--genre=ballet** remain randomised ...but at least you can be sure you’ll be listening to the Nutcracker or the Prince of the Pagodas and not Symphony No. 5 in C minor!

As with all the other parameters we're going to discuss next, Giocosio matches the tag values in a case-insensitive manner. If you tagged your music as "Opera", a selection for `--genre=opera` will match, even though one uses a capital 'O' and the other doesn't. The text match is also 'wild-carded on the outside', meaning that if you said `--genre=pe` ...that would match 'opera' just fine. If you said `--genre=on`, though, that would match both *concerto* and *symphony*. That would still work: you'd have limited your next music play to something that belongs to one or other of those two partly-matching genres. You are not now, however, being completely specific about a genre, but allowing some ambiguity to kick in - which can make for interesting listening, so go right ahead!

You cannot positively select multiple genres by supplying the `--genre` parameter more than once. If you do, the last value supplied is the one that will apply. Hence:

```
giocosio --dbname=main --genre=symphony --genre=ballet
```

...will result in a random selection of a ballet only for the next play, not symphonies. Naturally, as in the '`--genre=on`' example given earlier, if the supplied parameter is partially-spelled and thus ambiguous, permitting of a match to more than one genre, then yes: you could find music from any of the partial-match genres being selected for the next play.

10.3.6 Composer

In the same way the `--genre` parameter forces the selection of something to play from database records which match on the GENRE tag, so the `--composer` parameter allows you to force selection of things to play that have been tagged in the COMPOSER field with a particular matching value.

The parameter value is searched for regardless of case and is matched in a wild-carded manner. Thus a search for `--composer=brit` will find a match with Benjamin Britten easily enough. On the other hand, a search for `--composer=ga` will match both Havergal Brian and Edward Elgar -meaning that the next play could be of music by either of those fine composers. The fewer characters you type, therefore, the more possible matches there may be -and Giocosio will randomly select something to play from all those matches.

If you want to get very specific, such that only one, unambiguous composer's works are selected for the next random play, you may need to spell out the names with spaces in them. A search for `--composer=mozart`, for example, might get you Leopold or Wolfgang. So if you want to be very particular that it's only Leopold's music you want to listen to next, you'd need to spell out first and last names. As soon as you introduce spaces into the proceedings, however, you need to wrap the entire parameter value in double quotation marks. Thus, you would have to type;

```
giocosio --dbname=main --composer="Leopold Mozart"
```

...to get precise about it.

This rule of 'wrap arguments with spaces in quotation marks' is general, by the way: it applies to all parameters that take text values, not just the `--composer` one.

10.3.7 Performer

The **--performer** parameter is used to restrict Giocosio to playing only music whose PERFORMER tag matches (or partially matches) the parameter value. Bear in mind that the PERFORMER tag should, as per my *Axioms of Classical Tagging*, really only contain one name -and that name will be the ‘distinguishing artist’ name that makes *this* recording of a composition unique and distinguishable from *that* one. It won’t, therefore, usually include every person singing in an opera -but probably the fact that Georg Solti is conducting it.

If you want a quick way of saying ‘Giocosio, play me something that Karajan conducted’, therefore, the command:

```
giocosio --dbname=main --performer=karajan
```

...will likely do the job for you.

10.3.8 Comment

The **--comment** tag is used to make Giocosio find a random selection of music where the COMMENT tag contains something that matches the parameter value (as usual, the match is case-insensitive and wild-carded on the outside, meaning that a value of ‘cal’ will match Maria Callas and Calthorpe Burgundy equally well).

In my *Axioms of Classical Tagging*, I reserve the COMMENT tag for a free-form text description of all the artists performing on a recording. Don’t confuse this with the PERFORMER tag, which is reserved for the single name of the ‘distinguishing artist’. An opera may well have a conductor, an orchestra, a chorus and sixteen principle singers listed in COMMENT -but the PERFORMER tag would say (for example) just George Solti.

Keep the distinction in mind. When you are hoping to hear something Marissa Robles played harp on, chances are she’ll be listed in the COMMENT tag, not the PERFORMER one (she’s only the harpist, after all, not the orchestra conductor!)

Many people seem to dislike the way I suggest to use the COMMENT and PERFORMER tags this way, which is fair enough. So long as you know what data is stored in what tag, though, both parameters are available now to swing Giocosio to finding candidate plays that match whatever combination of parameter and values you now supply.

10.3.9 Composition

The **--composition** parameter allows you to say ‘Play Peter Grimes’ (as an example!):

```
giocosio --dbname=main --composition=grimes
```

The composition parameter value is compared in a wild-carded and case-insensitive way with the contents of the ALBUM tag, so you again might get more matches than you originally envisaged. If you’ve tagged Britten’s Sea Interludes as (for example) ‘Sea Interludes from Peter Grimes’, then the above command might well end up playing the orchestral Sea Interludes, not the opera you perhaps intended.

I want to pause at this point to mention that the five selection parameters I've just listed (genre, performer, composition, comment and composer) are all *exclusive*. If you try to use more than one at once, they won't work properly. Imagine asking for 'composer=beethoven and composition="Rite of Spring"', for example: the combination is illogical and cannot result in anything successfully matching both criteria. Allowing both parameters to be used simultaneously is therefore not permitted.

To avoid this sort of logical absurdity arising from multi-parameter use, Giocosio applies an 'order of precedence' to these particular parameters, as follows:

composition > composer > genre > performer > comment

That is, if you say "Composer=Britten and Comment=Callas", only the composer selection filter will apply: the comment filter will be silently ignored. Likewise, if you said "Composition=Lulu and Composer=Mozart", the composition takes precedence and you'll end up listening to the opera by Alban Berg. The requirement of the composer parameter is silently over-ridden by the higher-precedence composition one.

There is one exception to this general rule of 'one filter at a time', though: you are allowed to do a composer+genre combination filter, for those times when you specifically want to listen to a Beethoven symphony or a Haydn mass. So, genre+composer is respected in full, but any other combination of these selective filters will be regarded as merely a single filter, depending on the order of precedence. The genre+composer combo can also be negated (see below, Section 10.3.15), but only in its entirety. That is, if you say **--composer=haydn --genre=choral --negate** then you will making Giocosio look for not-choral compositions that are not-by Haydn. You cannot get it to apply the negation to just one of the filters, basically (so, 'I don't want Haydn, but I do want choral' is not achievable in this way).

10.3.10 Recordnumber

The **--recordnumber** parameter is an unusual one that takes a specific numeric value (unlike all the other selection parameters I've described before now, which all take textual entries).

The problem is that if I say '--composition="Peter Grimes"', I know I'm going to make Giocosio play *a* version of the Peter Grimes opera ...but I have several versions of that opera in my music collection, and a selection by composition name therefore doesn't tell me which one I'm going to get to listen to next.

If you happen to know that Britten's own recording of it is number 1862 in your collection and the Goodall mono version is number 1863, though, you can now say...

```
giocosio --dbname=main --recordnumber=1862
```

...and be guaranteed that it will be the Britten/Pears recording that gets played, not the Goodall.

This is an odd parameter, in fact, because it's the only one that completely and unambiguously removes all randomness from the music selection process: because each recording in your collection has a unique ID number assigned at the time the collection is scanned, you know without doubt precisely what recording is about to be played. It means you are in Random Play Mode (because you're using a database) but not actually randomising anything!

In fact, using `--recordnumber` is the exact equivalent of cd-ing to a specific music folder and launching Giocosio in Direct Play Mode there. It is as precisely deterministic. The difference, however, is that because you *are* in Random Play Mode, there's a database to record the play details into. A Direct Play Mode play of something would be forgotten the moment it's finished, because no record of it is kept anywhere -so it won't, for example, be able to be taken account of in relation to a time-bar calculation. But since we're still in Random Play Mode, but specifying a particular recording, the play event *will* be recorded in the database -and the fact you just played some Benjamin Britten at 9AM can now be taken account of when computing an applicable time-bar.

To use the `--recordnumber` parameter, you obviously need to know the precise IDs that each recording in your collection has been assigned in the Giocosio database. For that, you'll need to use the `--albumlist` report functionality, which is described in section 10.4.x

10.3.11 Unplayed

The `--unplayed` parameter searches for any **composers** that have not previously been played. It takes no arguments: the mere presence of the parameter on the command line launching Giocosio causes the player to find composers who do not have any records in the PLAYS table in the database.

Note the subtle difference between this parameter and the `--unplayedworks` one, which is described next.

10.3.12 Unplayedworks

The `--unplayedworks` parameter triggers Giocosio to search for any **compositions** which have not previously been played -where a "composition" is defined by the ALBUM tag and should (according to the *Axioms of Classical Tagging*) be the ordinary composition name *plus* the distinguishing artist's name *plus* the year of recording.

Note the subtle difference between `--unplayed` and `--unplayedworks`. Suppose you buy two new CDs. One an obscure oratorio by the hitherto unknown composer Walter von Braunenhoffenstop and the other a hot new recording of Beethoven's 5th Symphony, conducted by Theodore Currentnews.

Walter von Braunenhoffenstop is a brand new composer to you. You've not got anything else by him in your collection and you're desperate to hear the new acquisition. Running Giocosio with `--unplayed` is likely to trigger the playback of Walter's new oratorio, because Walter is a brand new and therefore *unplayed composer*.

But `--unplayed` will **not** help you hear your new recording of Beethoven. Beethoven's a composer you've probably got lots of other recordings of, and you've probably heard a bazillion versions of his various symphonies over the years. Beethoven is thus not an 'unplayed' composer... but that new recording of his Symphony No. 5 *is* an *unplayed composition*. Therefore, if your desire was to hear the new Beethoven recording, you'd want to run Giocosio with the `--unplayedworks` parameter.

The difference is subtle, but quite distinct.

Of course, Giocosio is always randomising -so, even if you said 'play me *unplayedworks*' there's no guarantee that Giocosio would choose *that particular new recording*, if you've got dozens of other

recordings to which you've never quite gotten around to listening -but at least the new Beethoven recording would be 'in with a chance'!

If you wanted to force the playing of that specific new recording, though: well, that's what the `--recordnumber` parameter is for. When random chance doesn't cut it, you can always force the playback of a specific recording with that.

10.3.13 Minduration

The `--minduration` parameter can be used in conjunction with any other parameter. It takes a numeric argument which represents minutes. The default value for this parameter -and the value that kicks in if you supply an invalid value for it, such as '-10', is 0. All recordings in your collection last at least 0 minutes, of course, so the default meaning of this parameter is to not exclude anything from playback.

The presence of this parameter with a valid, non-zero value makes Giocosio select randomly only those candidate recordings which are known to last for at least that number of **minutes**.

You can apply this parameter on its own or in conjunction with other parameters. Thus:

```
giocosio --dbname=main --minduration=30
```

...means 'play me something, anything, provided it lasts longer than 30 minutes'. Whereas:

```
giocosio --dbname=main --genre=ballet --minduration=30
```

...means 'play me a ballet that lasts longer than 30 minutes'. And so on.

10.3.14 Maxduration

The `--maxduration` parameter works exactly like the `--minduration` one, except that it sets a **maximum** length of time a piece of music can last for it to be a candidate for playing, with the parameter value again being a whole number of **minutes**.

Maxduration can be used on its own, in combination with other parameters -and even in combination with `--minduration`. This, for example, would be fine:

```
giocosio --dbname=main --genre=ballet --minduration=30 --maxduration=90
```

...and means 'play me a ballet that lasts between 30 and 90 minutes'.

The default value for `--maxduration` (and the value that kicks in if an invalid value is supplied) is 525960 minutes, which is about the number of minutes in a year -and thus effectively means 'no selection is being applied'.

If both min and maxduration are used at the same time, the minimum must be lower than the maximum... or the two parameters are simply ignored!

10.3.15 Negate

The `--negate` parameter is a bit of a strange one, in that it has no independent meaning, but merely reverses the sense of certain other parameters, if they happen to be present.

For example, **--comment=callas** means ‘find me something to play in which Maria Callas sings’. The parameters **--comment=callas --negate** simply mean, therefore, ‘find me something to play in which Maria Callas **doesn’t** sing! Similarly, **--composer=britten --negate** means ‘play me something by anyone other than Benjamin Britten’. And **--performer=karajan --negate** means ‘play me anything that isn’t being conducted by Karajan’.

The negate parameter works to reverse the meaning of the following other selective parameters:

- genre
- composer
- performer
- composition
- comment

It has no effect in relation to negating the meaning of any other parameters that might be present. You cannot, therefore, say **--minduration=10 --negate** as though you were trying to say ‘Play me things that do NOT have a minimum duration of 10 minutes’. That combination won’t work (and, surely, is logically equivalent to **--maxduration=10** in any case?!)

10.4 Reporting Parameters

These parameters allow Giocosos to produce reports about what it has played, what the recording collection looks like and so on: they thus have no meaning if Giocosos is only ever run in direct play mode, without a database.

<code>--report=xxxx</code>	An instruction to produce a report of the plays previously recorded within a database. Valid values for <i>xxx</i> are either a full path and filename for a written report, or the word 'screen' to display the report on-screen.	Requires the <code>--dbname</code> parameter, so we know what database to report on.
<code>--reportsort=y</code>	Orders the report by one of the columns it contains, where <i>y</i> is the column you want to be the sort column.	If absent, plays are reported in ascending chronological order (i.e., from earliest to latest)
<code>--reportdays=y</code>	Restricts the content of the report to plays made in the last <i>y</i> days	If absent, all plays are reported.
<code>--reporttype=xxxx</code>	Full or Diff(erential). Full is the default.	Export all plays, or only those plays made since the last export.
<code>--recordinglist</code>	An instruction to generate an on-screen list of all recordings, together with their unique ID numbers. Report is sorted by composer and within composer by composition name.	You need to know a recording's unique ID if you ever want to use the <code>--record-number</code> selection filter. Requires <code>--dbname</code> parameter so we know what database to query
<code>--composername=xxxx</code>	Used in conjunction with <code>--recordinglist</code> to restrict the on-screen list to the recordings attributed to a particular composer	The supplied parameter is wildcarded on the outside, so a value of '=brit' is sufficient to filter the recordings list to show only those composed by Benjamin Britten. Case-insensitive.
<code>--stats</code>	An instruction to produce a quick one-page statistical summary of the contents of the recording database, including numbers of composers, unplayed recordings, a distribution of recordings by their play length and so on.	

10.4.1. Report

The `--report` parameter triggers most of Giocosos's reporting capabilities and must be present for most of the other reporting parameters to make sense or function. It takes two possible arguments: a full path and filename if you want the report that's about to be generated written to a text file; or the word "screen" if you want the report displayed within the terminal session.

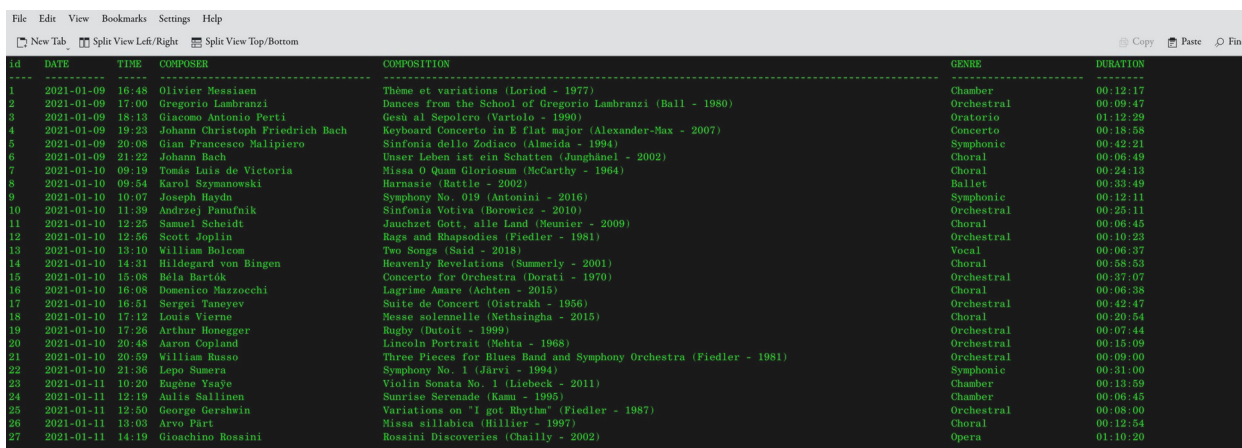
Note that the `--dbname` parameter must be present whenever `--report` is mentioned, because without it, we cannot know what database you want to query to generate the report in the first place. That database needs to be the same one you scanned when creating or refreshing the database, and the one you use when running Giocosos to play music.

The `--report` parameter always generates a listing of the plays that are recorded within the database. By default, it will report every single play that has been made, since the first time Giocosos was ever used to completely play anything.

The basic report can most simply be generated like this:

```
giocosos --report=screen --dbname=main
```

...and that will look like this:



ID	DATE	TIME	COMPOSER	COMPOSITION	GENRE	DURATION
1	2021-01-09	16:48	Olivier Messiaen	Thème et variations (Loriod - 1977)	Chamber	00:12:17
2	2021-01-09	17:00	Gregorio Lambranzi	Dances from the School of Gregorio Lambranzi (Ball - 1980)	Orchestral	00:09:47
3	2021-01-09	18:13	Giacomo Antonio Pertierra	Gesù al Sepolcro (Vartolo - 1990)	Oratorio	01:12:29
4	2021-01-09	19:23	Johann Christoph Friedrich Bach	Keyboard Concerto in E flat major (Alexander-Max - 2007)	Concerto	00:18:58
5	2021-01-09	20:08	Gian Francesco Malipiero	Sinfonia dello Zodiaco (Almeida - 1994)	Symphonic	00:42:21
6	2021-01-09	21:22	Johann Bach	Unser Leben ist ein Schatten (Junghnel - 2002)	Choral	00:06:49
7	2021-01-10	09:19	Tomás Luis de Victoria	Missa O Quam Gloriosum (McCarthy - 1964)	Choral	00:24:13
8	2021-01-10	09:54	Karol Szymanowski	Harnasie (Rattle - 2002)	Ballet	00:33:49
9	2021-01-10	10:07	Joseph Haydn	Symphony No. 019 (Antonini - 2016)	Symphonic	00:12:11
10	2021-01-10	11:39	Andrzej Panufnik	Sinfonia Votiva (Borowicz - 2010)	Orchestral	00:25:11
11	2021-01-10	12:25	Samuel Scheidt	Jauchzet Gott, alle Land (Wemmer - 2009)	Choral	00:08:45
12	2021-01-10	12:56	Scott Joplin	Rags and Rhapsodies (Fiedler - 1981)	Orchestral	00:10:23
13	2021-01-10	13:10	William Bolcom	Two Songs (Said - 2018)	Vocal	00:06:37
14	2021-01-10	14:31	Hildegard von Bingen	Heavenly Revelations (Summerly - 2001)	Choral	00:58:53
15	2021-01-10	15:08	Béla Bartók	Concerto for Orchestra (Dorati - 1970)	Orchestral	00:37:07
16	2021-01-10	16:08	Domenico Mazzocchi	Lagrime Amare (Achten - 2015)	Choral	00:06:38
17	2021-01-10	16:51	Sergei Prokofiev	Suite de Concert (Oistrakh - 1956)	Orchestral	00:42:47
18	2021-01-10	17:12	Louis Vierne	Messe solennelle (Nethsingha - 2015)	Choral	00:20:54
19	2021-01-10	17:36	Arthur Honegger	Rugby (Dutoit - 1999)	Orchestral	00:07:44
20	2021-01-10	20:48	Aaron Copland	Lincoln Portrait (Mehta - 1968)	Orchestral	00:15:09
21	2021-01-10	20:59	William Russo	Three Pieces for Blues Band and Symphony Orchestra (Fiedler - 1981)	Orchestral	00:09:00
22	2021-01-10	21:38	Leopold Stokowski	Symphony No. 1 (Jarvi - 1994)	Symphonic	00:31:08
23	2021-01-11	10:20	Eugene Ysaÿe	Violin Sonata No. 1 (Liebeck - 2011)	Chamber	00:13:59
24	2021-01-11	12:19	Aulis Sallinen	Sunrise Serenade (Kann - 1995)	Chamber	00:06:45
25	2021-01-11	12:50	George Gershwin	Variations on "I got Rhythm" (Fiedler - 1987)	Orchestral	00:08:00
26	2021-01-11	13:03	Arvo Pärt	Missa sillabica (Hillier - 1997)	Choral	00:12:54
27	2021-01-11	14:19	Gioacchino Rossini	Rossini Discoveries (Chailly - 2002)	Opera	01:10:20

You'll notice that it consists of six columns of meaningful data, plus an initial 'ID' column which is simply a unique identifier for each row that increments from 1 upwards, without limit. You can also regard the date and time columns as being part of a single 'date and time of play completion' column. So that means there are really only five columns of 'meaningful' data -and, by default, the report is produced sorting by the date/time combo one: you can see my plays are listed starting on January 9th 2021, incrementing steadily by day thereafter.

That sort-ordering is not much use, however, if you want to ask 'how many times have I played Anton Bruckner?' That is why you may want to generate a report with some additional parameters, such as `--reportsort` and `--reportdays`, which I shall discuss next.

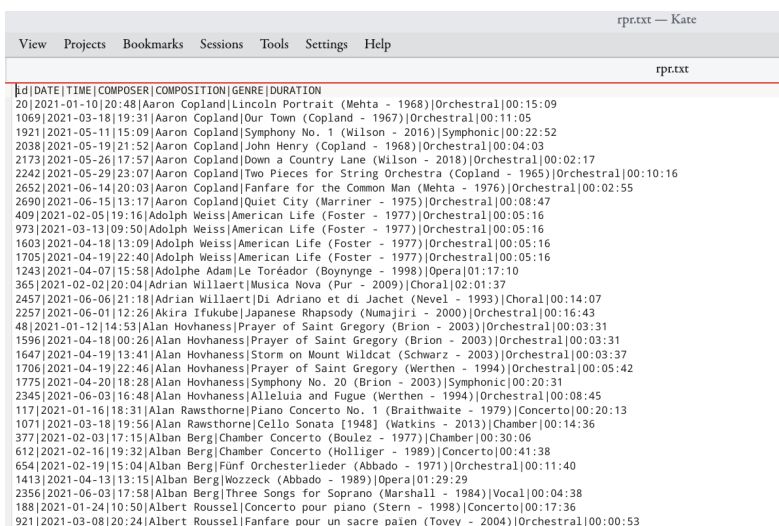
If a report is displayed on screen, it can be scrolled through one line at a time in either direction using the Up- and Down-arrow keys; you can also page through the report in either direction, a screen at a time, using the PgUp and PgDn keys.

To stop displaying the report on screen, just tap the letter ‘q’ (for ‘quit’).

If you direct the report to a path/filename combo, rather than to ‘screen’, you’ll be able to open it in any text editor. This command:

```
giocososo --report=$HOME/Desktop/rpr.txt --dbname=main --reportsort=2
```

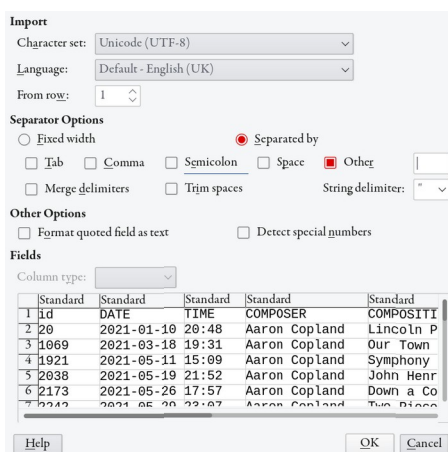
...produces something that is very similar to that seen in the previous screenshot, but won’t look anything like it! Here is a sample:



id	DATE	TIME	COMPOSER	COMPOSITION	GENRE	DURATION
20	2021-01-10	20:48	Aaron Copland	Lincoln Portrait (Mehta - 1968)	Orchestral	00:15:09
1069	2021-03-18	19:31	Aaron Copland	Our Town (Copland - 1967)	Orchestral	00:11:05
1921	2021-05-11	15:09	Aaron Copland	Symphony No. 1 (Wilson - 2016)	Symphonic	00:22:52
2038	2021-05-19	21:52	Aaron Copland	John Henry (Copland - 1968)	Orchestral	00:04:03
2173	2021-05-26	17:57	Aaron Copland	Down a Country Lane (Wilson - 2018)	Orchestral	00:02:17
2242	2021-05-29	23:07	Aaron Copland	Two Pieces for String Orchestra (Copland - 1965)	Orchestral	00:10:16
2652	2021-06-14	20:03	Aaron Copland	Fanfare for the Common Man (Mehta - 1976)	Orchestral	00:02:55
2690	2021-06-15	13:17	Aaron Copland	Quiet City (Marriner - 1975)	Orchestral	00:08:47
409	2021-02-05	19:16	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16
973	2021-03-13	09:50	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16
1603	2021-04-18	13:09	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16
1705	2021-04-19	22:40	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16
1243	2021-04-07	15:58	Adolphe Adam	Le Toréador (Boynynge - 1998)	Opera	01:17:10
365	2021-02-02	20:04	Adrian Willaert	Musica Nova (Pur - 2009)	Choral	02:01:37
2457	2021-06-06	21:18	Adrian Willaert	Di Adriano et di Jachet (Nevel - 1993)	Choral	00:14:07
2257	2021-06-01	12:26	Akira Ifukube	Japanese Rhapsody (Numajiri - 2000)	Orchestral	00:16:43
48	2021-01-12	14:53	Alan Hovhaness	Prayer of Saint Gregory (Brion - 2003)	Orchestral	00:03:31
1596	2021-04-18	00:26	Alan Hovhaness	Prayer of Saint Gregory (Brion - 2003)	Orchestral	00:03:31
1647	2021-04-19	13:41	Alan Hovhaness	Storm on Mount Wildcat (Schwarz - 2003)	Orchestral	00:03:37
1706	2021-04-19	22:46	Alan Hovhaness	Prayer of Saint Gregory (Werthen - 1994)	Orchestral	00:05:42
1775	2021-04-20	18:28	Alan Hovhaness	Symphony No. 20 (Brion - 2003)	Symphonic	00:20:31
2345	2021-06-03	16:48	Alan Hovhaness	Alleluia and Fugue (Werthen - 1994)	Orchestral	00:08:45
117	2021-01-16	18:31	Alan Rawsthorne	Piano Concerto No. 1 (Braithwaite - 1979)	Concerto	00:20:13
1071	2021-03-18	19:56	Alan Rawsthorne	Cello Sonata [1948] (Watkins - 2013)	Chamber	00:14:36
377	2021-02-03	17:15	Alban Berg	Chamber Concerto (Boulez - 1977)	Chamber	00:30:06
612	2021-02-16	19:32	Alban Berg	Chamber Concerto (Holliger - 1989)	Concerto	00:41:38
654	2021-02-19	15:04	Alban Berg	Fünf Orchesterlieder (Abbado - 1971)	Orchestral	00:11:40
1413	2021-04-13	13:15	Alban Berg	Wozzeck (Abbado - 1989)	Opera	01:29:29
2356	2021-06-03	17:58	Alban Berg	Three Songs for Soprano (Marshall - 1984)	Vocal	00:04:38
188	2021-01-24	10:50	Albert Roussel	Concerto pour piano (Stern - 1998)	Concerto	00:17:36
921	2021-03-08	20:24	Albert Roussel	Fanfare pour un sacre païen (Tovey - 2004)	Orchestral	00:00:53

The thing which makes this version of the report look weird is that every piece of data is separated from every other piece by a ‘pipe character’ (i.e., |). That makes this, therefore, a ‘pipe-delimited text file’, which is very similar to a ‘comma-separated variable file’ (CSV), except that it’s a pipe character doing the delimiting, not a comma.

The use of a different delimiter to what you may be used to makes little difference to your ability to import these sorts of files into tools such as Excel or LibreOffice Calc. Here, for example, I’m trying to open the file in LibreOffice’s spreadsheet program (Calc):



Import

Character set: Unicode (UTF-8)

Language: Default - English (UK)

From row: 1

Separator Options

☐ Fixed width ☒ Separated by

☐ Tab ☐ Comma ☐ Semicolon ☐ Space ☒ Other |

☐ Merge delimiters ☐ Trim spaces String delimiter: *

Other Options

☒ Format quoted field as text ☐ Detect special numbers

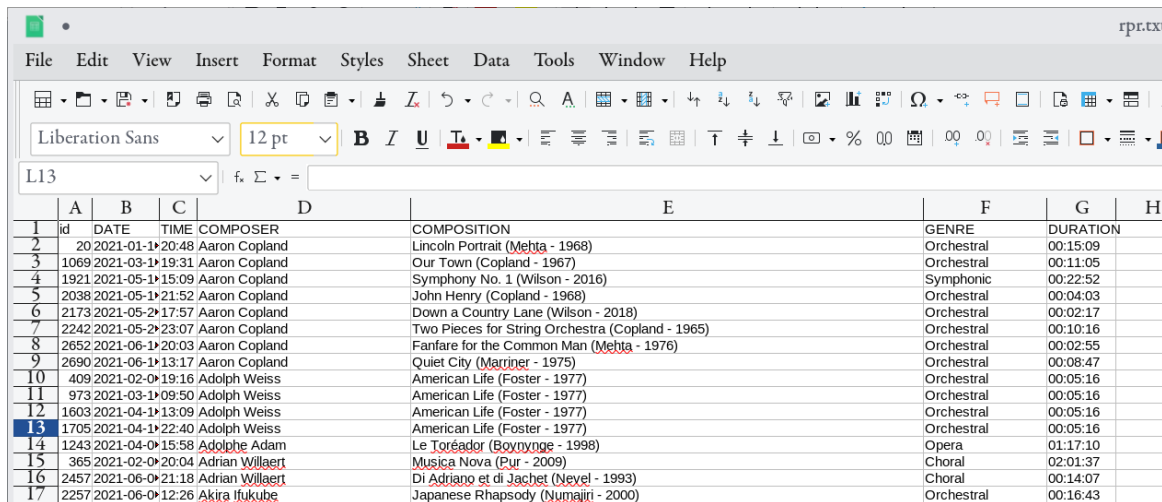
Fields

Column type: Standard

	Standard	Standard	Standard	Standard	Standard
1	id	DATE	TIME	COMPOSER	COMPOSITION
2	20	2021-01-10	20:48	Aaron Copland	Lincoln P
3	1069	2021-03-18	19:31	Aaron Copland	Our Town
4	1921	2021-05-11	15:09	Aaron Copland	Symphony
5	2038	2021-05-19	21:52	Aaron Copland	John Henr
6	2173	2021-05-26	17:57	Aaron Copland	Down a Co
7	2242	2021-05-29	23:07	Aaron Copland	Two Piece

Help OK Cancel

I simply tell it that the file is ‘Separated by... Other’ and type a pipe character into the field next to the word ‘other’. Immediately, the file preview shows it interpreting the data correctly into distinct columns of data. The final import is then easy:



	A	B	C	D	E	F	G	H
1	id	DATE	TIME	COMPOSER	COMPOSITION	GENRE	DURATION	
2	20	2021-01-1*	20:48	Aaron Copland	Lincoln Portrait (Mehta - 1968)	Orchestral	00:15:09	
3	1069	2021-03-1*	19:31	Aaron Copland	Our Town (Copland - 1967)	Orchestral	00:11:05	
4	1921	2021-05-1*	15:09	Aaron Copland	Symphony No. 1 (Wilson - 2016)	Symphonic	00:22:52	
5	2038	2021-05-1*	21:52	Aaron Copland	John Henry (Copland - 1968)	Orchestral	00:04:03	
6	2173	2021-05-2*	17:57	Aaron Copland	Down a Country Lane (Wilson - 2018)	Orchestral	00:02:17	
7	2242	2021-05-2*	23:07	Aaron Copland	Two Pieces for String Orchestra (Copland - 1965)	Orchestral	00:10:16	
8	2652	2021-06-1*	20:03	Aaron Copland	Fanfare for the Common Man (Mehta - 1976)	Orchestral	00:02:55	
9	2690	2021-06-1*	13:17	Aaron Copland	Quiet City (Marriner - 1975)	Orchestral	00:08:47	
10	409	2021-02-0*	19:16	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16	
11	973	2021-03-1*	09:50	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16	
12	1603	2021-04-1*	13:09	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16	
13	1705	2021-04-1*	22:40	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16	
14	1243	2021-04-0*	15:58	Adolphe Adam	Le Toréador (Boismyng - 1998)	Opera	01:17:10	
15	365	2021-02-0*	20:04	Adrian Willaert	Musica Nova (Pur - 2009)	Choral	02:01:37	
16	2457	2021-06-0*	21:18	Adrian Willaert	Di Adriano et di Jachet (Nevel - 1993)	Choral	00:14:07	
17	2257	2021-06-0*	12:26	Akira Ifukube	Japanese Rhapsody (Numajiri - 2000)	Orchestral	00:16:43	

Once you can get Giocososo data into a spreadsheet in this way, it’s trivial to be able to generate graphs and pie-charts from it, helping you to analyse and understand your music listening habits.

10.4.3 Reportsort

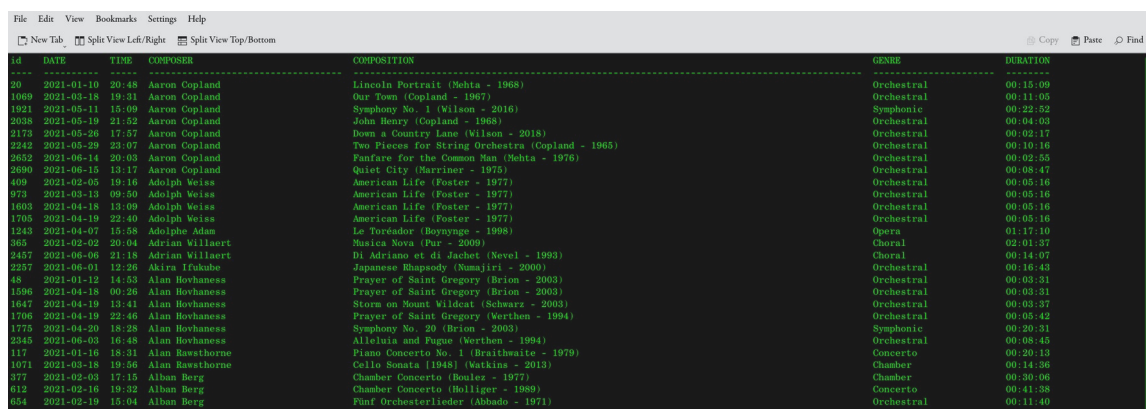
As you’ve just seen, the on-screen report can be thought of as consisting of 5 meaningful columns of data. Label the date/time one ‘1’ and go rightwards, incrementing as each new column appears: that means the COMPOSER column is ‘2’, the COMPOSITION=’3’, the GENRE is column ‘4’ and, finally, the DURATION column is ‘5’.

The **--reportsort** parameter takes a numeric value, from 1 to 5, indicating which of those columns you want to sort the report by. The default, as you’ve just seen, is to sort by the date/time combo column, so **--reportsort=1** would appear identical to the previous screenshot.

This command, however:

```
giocososo --report=screen --dbname=main --reportsort=2
```

....produces a report that looks like this:



id	DATE	TIME	COMPOSER	COMPOSITION	GENRE	DURATION
20	2021-01-10	20:48	Aaron Copland	Lincoln Portrait (Mehta - 1968)	Orchestral	00:15:09
1069	2021-03-18	19:31	Aaron Copland	Our Town (Copland - 1967)	Orchestral	00:11:05
1921	2021-05-11	15:09	Aaron Copland	Symphony No. 1 (Wilson - 2016)	Symphonic	00:22:52
2038	2021-05-19	21:52	Aaron Copland	John Henry (Copland - 1968)	Orchestral	00:04:03
2173	2021-05-26	17:57	Aaron Copland	Down a Country Lane (Wilson - 2018)	Orchestral	00:02:17
2242	2021-05-29	23:07	Aaron Copland	Two Pieces for String Orchestra (Copland - 1965)	Orchestral	00:10:16
2652	2021-06-14	20:03	Aaron Copland	Fanfare for the Common Man (Mehta - 1976)	Orchestral	00:02:55
2690	2021-06-15	13:17	Aaron Copland	Quiet City (Marriner - 1975)	Orchestral	00:08:47
409	2021-02-05	19:16	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16
973	2021-03-15	09:50	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16
1603	2021-04-18	13:09	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16
1705	2021-04-19	22:40	Adolph Weiss	American Life (Foster - 1977)	Orchestral	00:05:16
1243	2021-04-07	15:58	Adolphe Adam	Le Toréador (Boismyng - 1998)	Opera	01:17:10
365	2021-02-02	20:04	Adrian Willaert	Musica Nova (Pur - 2009)	Choral	02:01:37
2457	2021-06-06	21:18	Adrian Willaert	Di Adriano et di Jachet (Nevel - 1993)	Choral	00:14:07
2257	2021-06-01	12:26	Akira Ifukube	Japanese Rhapsody (Numajiri - 2000)	Orchestral	00:16:43
48	2021-01-12	14:53	Alan Hovhanness	Prayer of Saint Gregory (Brion - 2003)	Orchestral	00:03:31
1586	2021-04-15	00:26	Alan Hovhanness	Prayer of Saint Gregory (Brion - 2003)	Orchestral	00:03:31
1647	2021-04-19	13:41	Alan Hovhanness	Storm on Mount Wildcat (Schwarz - 2003)	Orchestral	00:03:37
1706	2021-04-19	22:46	Alan Hovhanness	Prayer of Saint Gregory (Werthen - 1994)	Orchestral	00:05:42
1775	2021-04-20	18:28	Alan Hovhanness	Symphony No. 20 (Brion - 2003)	Symphonic	00:20:31
2245	2021-06-03	16:48	Alan Hovhanness	Alleluia and Pange (Werthen - 1994)	Orchestral	00:08:45
117	2021-01-16	18:31	Alan Rawsthorne	Piano Concerto No. 1 (Braithwaite - 1979)	Concerto	00:20:13
1071	2021-03-18	19:56	Alan Rawsthorne	Cello Sonata [1948] (Watkins - 2013)	Chamber	00:14:36
377	2021-02-03	17:15	Alban Berg	Chamber Concerto (Boulez - 1977)	Chamber	00:30:08
312	2021-02-16	19:32	Alban Berg	Chamber Concerto (Boulez - 1980)	Chamber	00:41:36
654	2021-02-19	15:04	Alban Berg	Pinf Orchesterlieder (Abbado - 1971)	Orchestral	00:11:40

...and here you'll see all the Aaron Copland is listed together, followed by all the Adolph Weiss and so on. Here, then, you're now reporting all the plays you've ever made using Giocosio by the composer column.

If you were therefore ever wondering, 'When was the last time I ever play *Das Lied von der Erde*?', you'd probably best run `--report --reportsort=3`, thereby ensuring all the recordings/compositions were sorted together by their name. Scroll quickly down to the 'D' entries, and pretty soon, you'd see all your *Das Leid* plays listed together:

ID	Date	Time	Composer	Title	Instrumentation	Duration
322	2021-02-11	23:49	Zoltan Kodaly	Dances of Galanta (Fischer - 1964)	Orchestral	00:15:53
3119	2021-06-21	18:21	Diderik Buxtehude	Danket dem Herren, denn er ist sehr freundlich BuxWV181 (Stella - 2011)	Keyboard	00:03:27
688	2021-02-20	15:00	Rued Langgaard	Danmarks Radio - Fanfares for Orchestra (Langgaard - 2004)	Orchestral	00:01:18
992	2021-03-14	20:21	Rued Langgaard	Danmarks Radio - Fanfares for Orchestra (Langgaard - 2004)	Orchestral	00:01:18
2218	2021-05-29	17:53	Claude Debussy	Danse Sacrée et Danse Profane (Marrinon - 1975)	Orchestral	00:09:22
3371	2021-06-25	10:53	Claude Debussy	Danse (Tarentelle styrienne) (Ravouzet - 2007)	Keyboard	00:04:45
3153	2021-06-22	09:18	Claude Debussy	Danse bohémienne (Ravouzet - 2008)	Keyboard	00:02:03
428	2021-02-08	18:28	Claude Debussy	Danse bohémienne (Ras - 1998)	Keyboard	00:01:55
461	2021-02-08	15:37	Henriette Renié	Danse des Latins (Loman - 1998)	Chamber	00:04:00
1316	2021-04-09	21:15	Henriette Renié	Danse des Latins (Loman - 1998)	Chamber	00:04:00
1490	2021-04-15	12:07	Henriette Renié	Danse des Latins (Loman - 1998)	Chamber	00:04:00
1561	2021-04-16	20:57	Henriette Renié	Danse des Latins (Loman - 1998)	Chamber	00:04:00
1690	2021-04-19	21:31	Henriette Renié	Danse des Latins (Loman - 1998)	Chamber	00:04:00
2440	2021-06-06	07:58	Henriette Renié	Danse des Latins (Loman - 1998)	Chamber	00:04:00
1955	2021-05-13	15:09	Boris Tishchenko	Dante Symphony No. 3 - Inferno (Alekseev - 2009)	Symphonic	00:39:37
2487	2021-06-11	21:30	Richard Strauss	Daphne Etude (Sebastyen - 1995)	Chamber	00:01:43
842	2021-02-03	19:29	Jean-Philippe Rameau	Daphnis et Egle (Tercy-Smith - 1997)	Orchestral	00:29:08
1585	2021-04-17	18:35	Edwy L. Roman	Darktown Strutter's Ball (Masur - 1974)	Orchestral	00:03:27
244	2021-01-26	20:05	Johann Michael Bach	Das Blut Jesu Christi (Junghänel - 2002)	Choral	00:03:26
1272	2021-04-08	20:57	Gustav Mahler	Das Knaben Wunderhorn (Tourel - 2001)	Vocal	00:02:52
149	2021-01-21	14:41	Gustav Mahler	Das Lied von der Erde (Bernstein - 1972)	Symphonic	01:03:41
495	2021-02-10	22:55	Gustav Mahler	Das Lied von der Erde (Klimpfer - 1964)	Symphonic	01:04:06
2983	2021-06-20	12:51	Richard Strauss	Das Schloß am Meer (Neonovius - 1986)	Keyboard	00:05:57
2715	2021-06-15	18:28	Sammel Scheidt	Das alte Jahr vergangen ist (Neunier - 2009)	Choral	00:05:44
3324	2021-06-24	18:06	Ralph Vaughan Williams	Dawn Patrol (Groves - 2008)	Orchestral	00:04:11
2369	2021-06-04	16:30	Nedra Wassergsky	Down on the Sacore River (Bassham - 1991)	Orchestral	00:09:51
2645	2021-06-14	19:35	Frank Bridge	Day after day (Riches - 2004)	Orchestral	00:04:55
2746	2021-06-16	10:05	Krzysztof Penderecki	De natura sonoris II (Wit - 2000)	Orchestral	00:08:58
2870	2021-06-18	18:35	Alexander Agricola	De tous biens playne (Nevel - 1998)	Choral	00:02:15

...so now I know I last played it on 10th February 2021, finishing at around 11pm!

Having once generated a report in one sort order, you cannot then change the ordering, short of pressing 'q' to quit and then relaunching the report with a different `--reportsort` number.

10.4.2 Reportdays

By default, all reports generated by the `--report` parameter, however they may be sorted, will include every play made in Giocosio since you started using Giocosio. As time passes, that makes for ever-longer reports, of course. The `--reportdays` parameter therefore accepts a numeric value that limits the report to listing only those plays which have happened in the last *y* days.

The default number of dsays is 36,500 -which, of course, is the number of days in a century and thus is functionally equivalent to 'no reportdays filter applied at all' If you supply a non-valid number of days for this parameter (for example, a negative number of days, or a non-integer value such as '10.5'), that default value is silently applied in the invalid value's place.

10.4.4 Reporttype

I've already mentioned that a report, whether directed to screen or text, will contain -by default- the complete history of plays made with Giocosio or, if that gets too big, can be filtered by a number of days. Filtering by day can be a bit awkward though, as it's not terribly precise. For example, you played some music 7 days ago at 9.25AM. You run a `--reportdays=7` report at 9.22AM and that play will be included on the report, because it took place *within* the previous 7 days. Run the exact same report 4 minutes later, however, and that play will disappear off the report... because now `--reportdays=7` takes you back to plays made since 9:26AM 7 days ago... and that play is now one minute past its sell-by date.

Thus, Giocososo allows you to specify a different way of saying what should be on the report or not: **--reporttype** can be supplied with a value of either “full” or “diff” -and a differential report is guaranteed to contain only those records which were not included on the previous differential report.

You’ve already seen that every play in Giocososo’s database is assigned a meaningless and invariant ‘ID number’ to uniquely identify it. Giocososo happens also to record in its database the highest ID number it has previously exported in a differential export. So, if you run a ‘diff’ report on Monday, it might include plays up to 3017; if you run a second ‘diff’ report on Tuesday, it will only include plays from 3018 onwards. (The highest number reached in each report export is stored in the MUSICPLAYS table, in a column called MAXPLAYEXPORT).

Note that a ‘full’ report (which is the default report type) doesn’t alter the maxplayexport number stored in the database, so you can do a series of differential exports, followed by a full... and the next differential will pick up the plays done since the last differential export: the full export done in between won’t have altered the ‘play number reached’ at all.

It has to be said that the use of differential reporting is pretty specialised and it’s not something a lot of people will want to do very often, if ever. As an example of what’s possible with it, however, take a look at my own website (at <https://absolutelybaching.com/music-plays/>): the list of my plays shown there is derived directly as a report from Giocososo on my main PC. Rather than export every play since the dawn of 2021, however, and wiping the existing data and replacing it every time, I instead produce a differential export every 15 minutes out of Giocososo and ship what amounts to a list of ‘the latest plays’ to the website, where it’s added to the records that already exist.

This sort of reporting on your music listening habits is not going to appeal to everyone, of course: but Giocososo supports it, even so.

10.4.5 Recordinglist

The **--report**, **--reportsort**, **--reportdays** and **--reporttype** parameters are all associated with producing lists of what *plays* have been made by Giocososo. How about reporting on what *recordings* are available to Giocososo? That’s where the **--recordinglist** parameter comes in. Run that (with a **--dbname** parameter) and suddenly Giocososo will tell you about all the compositions it knows of. The parameter takes no argument: it’s either present or not. If it is, you’ll see this sort of output on screen:

id	COMPOSER	COMPOSITION
16	Aaron Copland	An Outdoor Overture (Copland - 1969)
17	Aaron Copland	An Outdoor Overture (Wilson - 2016)
1	Aaron Copland	Appalachian Spring (Bernstein - 1961)
2	Aaron Copland	Appalachian Spring (Copland - 1970)
3	Aaron Copland	Appalachian Spring (Mehta - 1976)
4	Aaron Copland	Appalachian Spring (Slatkin - 2014)
5	Aaron Copland	Billy the Kid (Copland - 1969)
18	Aaron Copland	Billy the Kid Suite (Bernstein - 1959)
19	Aaron Copland	Ceremonial Fanfare (Jones - 1976)
10	Aaron Copland	Clarinet Concerto (Bernstein - 1991)
11	Aaron Copland	Clarinet Concerto (Copland - 1963)
12	Aaron Copland	Clarinet Concerto (Singer - 2008)
21	Aaron Copland	Connotations (Wilson - 2018)
20	Aaron Copland	Connotations for Orchestra (Copland - 2004)
48	Aaron Copland	Dance Symphony (Copland - 1967)

It's a very simple report, listing compositions (or 'recordings' if you like) in composer/composition name order. So everything by Aarton Copland comes before everything by Benjamin Britten, and within Benjamin Britten, *Peter Grimes* comes before *Young Apollo*.

This explains the apparent odd and jumbled appearance of the ID column: it's not sequential, because the report is not listing things in ID order.

The point of this report is simply to allow you to associate rapidly the unique ID with a particular recording -and you might need to do that if you ever want to tell Giocososo to play, specifically, Copland's own 1969 recording of his Outdoor Overture. If you want to do that, you'll need to know that that specific recording has, uniquely in Giocososo's database, the ID number 18. Knowing that, you can then issue the command:

```
giocososo --recordnumber=18
```

...and that particular recording will then be played, to the exclusion of all others.

As previously described in **Section 10.3.10**, in other words, whenever you want to stop using Giocososo's random play capabilities and instead insist on a deterministic play of a specific recording of a composition, you'll use `--recordnumber` to achieve that... and `--recordinglist` gives you the information you need to feed that parameter with the required unique ID number.

10.4.6 Composername

The `--composername` parameter is used only in association with the `--recordinglist` parameter. You've already seen why you might want to produce a recordinglist report... and that it starts listing things by composers in A, B, C order when you do so.

If you were wanting to know the unique ID associated with Zbigniew Preisner's *Requiem for my friend*, though, it might be a bit annoying to have to page through every single recording you own, sorted by the Aaron Coplands, the Benjamin Brittens, the Camille Saint-Saëns and so on, first.

A large music collection can take a long time to page through, when sorted by composer name, basically. For that reason, the `--composername=xxx` parameter is provided so that you can filter the `--recordinglist` report by a particular composer's name or part-name.

To hunt down that piece by Zbigniew Preisner, for example, this command will help:

```
giocososo --recordinglist --composername=preis --dbname=main
```

```
sorted by the Aaron Coplands, the Benjamin Brittens, the Camille Saint-Saëns and so on, first. For that reason, the --composername=xxx parameter is provided. Use it to supply a composer's name (or the COMPOSER) will filter the a COMPOSITION accordingly.
-----
This command would do the job, for example:
12721 Zbigniew Preisner Diaries of Hope (Klocek - 2013)
12720 Zbigniew Preisner Requiem for my Friend (Kaspszyk - 1998)
(END)
```

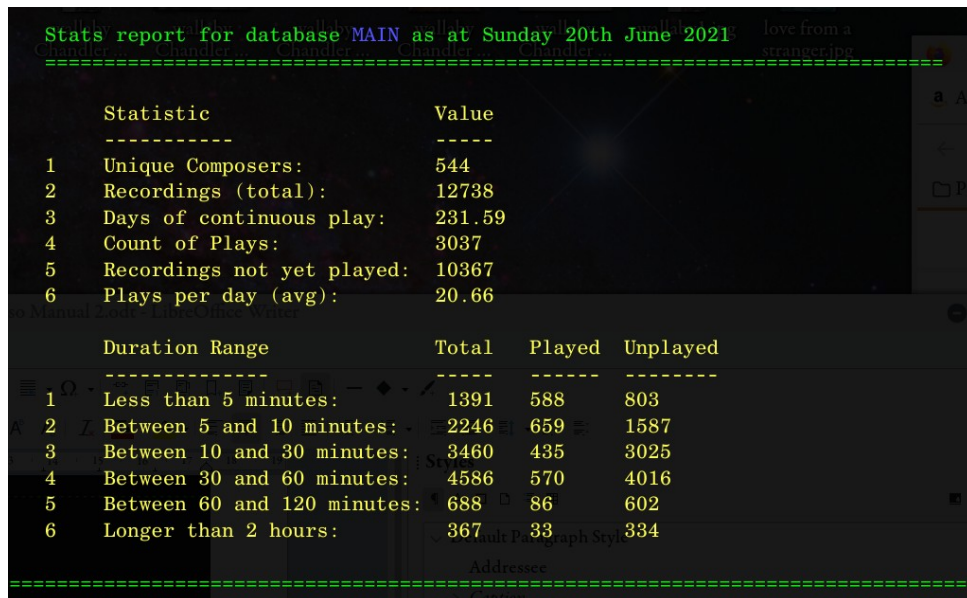
...and now only recordings by someone called '...preis...'-something-or-other are listed and I can immediately see that I should be asking Giocososo to play `--recordnumber=12720` to hear the Requiem I

wanted. As with all these textural-style parameters, therefore, note that the `--composername` parameter value is wild-carded on the outside and is handled case-insensitively.

10.4.7 Stats

The last reporting parameter to mention is the `--stats` one. It takes no value, but its presence will make Giocosio display a quick one-page summary report of the state of your music collection. The parameter needs the `--dbname` parameter to tell it what database to query for the statistics figures. So this command will do the job, for example:

```
giocosio --dbname=main --stats
```



```
Stats report for database MAIN as at Sunday 20th June 2021
=====
```

	Statistic	Value
1	Unique Composers:	544
2	Recordings (total):	12738
3	Days of continuous play:	231.59
4	Count of Plays:	3037
5	Recordings not yet played:	10367
6	Plays per day (avg):	20.66

	Duration Range	Total	Played	Unplayed
1	Less than 5 minutes:	1391	588	803
2	Between 5 and 10 minutes:	2246	659	1587
3	Between 10 and 30 minutes:	3460	435	3025
4	Between 30 and 60 minutes:	4586	570	4016
5	Between 60 and 120 minutes:	688	86	602
6	Longer than 2 hours:	367	33	334

```
=====
```

These statistics tell me, in the upper half, how many unique recordings I have, and by how many unique composers. I can see the ‘continual play length’ of my music collection: I have enough music here to play continuously for 231½ days, for example. I can see at a glance how many of my recordings I’ve actually played (only 3,037 of the 12,788 available to me), with over 10,300 recordings never having been played. (Don’t try to derive one of these figures from the others, by the way: they don’t work like that! If I play the same recording 500 times, it reduces the recordings not-yet-played figure by 1, because it’s a single recording, but increases the number of plays by 500, because each play ‘counts’!)

In the bottom half of the report, you get a sort-of histogram distribution report: these numbers should add up to the ‘Recordings (total)’ figure displayed in the top-half of the report. You can see I have an alarmingly high number of sub-5 minute recordings, which might indicate that I should re-think the way I have catalogued lots of little snippets of music as ‘recordings’ in their own right. But you can also see that the vast bulk of my music collection lasts between 10 and 60 minutes, which seems quite reasonable. The ‘more than 2 hours’ recordings are probably mostly Wagner ones!

For each ‘duration band’, you also get played v. unplayed figures -and these do add up, so 588+803 in the first row should (and does!) equal the 1391 total of sub-5 minute recordings I’ve got in my collection.

Finally, note that you cannot combine the three sorts of report that Giocososo can produce. Ask for `--recordinglist` and `--stats`, for example, and you'll only get the Album List. Ask for a `--report` and an `--recordinglist` and you'll only get a Plays Report. And so on: think of these as three distinct types of report and choose to produce only one of them at a time and you'll be fine. Their order of precedence is: *Report* > *Recordinglist* > *Stats*.

10.5 Run-time Parameter Summary

I suppose it goes without saying -but I'll say it anyway: this sort of reporting (and, indeed, this sort of filtering of play selections) can only take place meaningfully in the presence of an extremely well-organised music collection, where tags have been applied consistently and the physical location of your music closely and faithfully reflects the logical ordering of your metadata choices.

In a world where Beethoven could be catalogued as "Beethoven", "Beethoven, L", "Beethoven, L. v.", "Ludwig van Beethoven" or "Ludwig Beethoven", you are going to struggle to output meaningful reports of how much Beethoven you own, let alone persuade Giocososo to play it to the exclusion of all other music. If your data describes a single entity in 15 different ways, nothing can help you make coherent sense of that, really!

Hopefully, you'll see from this tour of the various run-time parameters that Giocososo can use that Giocososo is capable of very subtle manipulation: 'play me things by Britten that were ballets that don't take more than 40 minutes to play', for example. I don't know any other music players that can do that sort of thing... but Giocososo can only do it if your metadata tags correctly and consistently describe Britten and Ballet!

Giocososo is flexible, for sure; but fundamentally, it's only as good as your metadata!

11.0 Some Technical Extras

11.1 Use of ALSA

Giocosos is intended to output its audio signal to an audio device. The process of deducing *which* audio device hardware address to direct output to was already described above, in [Section 10.2.4](#).

Usually, Giocosos will use ALSA to output sound. Ideally, Giocosos's sound output will not go via Pulseaudio at all. This is a deliberate design decision: natively, ALSA can only output one sound event at a time. If your music player has 'hogged' the ALSA audio device, then no other audio can be heard whilst music plays (which is obviously desirable in a classical music player). Conversely, it is possible to visit (say) Youtube and begin playback of something with an audio track -and, at this point, the audio device cannot be accessed by Giocosos. Even if you pause video playback in your web browser, it has 'dibbs' on the audio device and no other audio can make it through. Music may appear to play in Giocosos, but only silence results.

As mentioned, this behaviour is by design -but it may not to be everyone's taste. If your distro uses Pulseaudio (which can do device sharing out of the box), you may never even notice this behaviour.

The topic of audio on Linux is extremely complicated, however, and well out of scope for this document. I merely point out that you may want to make sure that Giocosos is the only program outputting -or even *trying* to output- audio at all times. If extended silence results when Giocosos claims it is playing something, the problem is either in the choice of output device (as explained in [Section 10.2.4](#)) or that the output device is an ALSA one that has already had audio directed at it which has not yet been told to 'let go' of the device.

11.2 Lost Scrobbles

Giocosos makes every effort to transmit the details of the most recent play to Last.fm if scrobbling has been (a) configured [see [Section 8.1](#)] and (b) enabled [see [Section 8.2](#)].

However, Last.fm is an external website and may well be beset at various times of the day with administrative outages, denial of service attacks and any number of other problems that prevents anyone connecting to it.

If this network 'outage' occurs at the point where Giocosos want to scrobble, there will be lengthier-than-normal pause between plays in a cycle of selections, and the 'Now scrobbling' message may be appended with a red 'Error!' message.

At this point, Giocosos simply gives up trying to scrobble the last-played tracks and moves on to playing the next (if the number of --selections is not yet satisfied, anyway).

The failed scrobble is 'lost' and cannot be re-captured or re-submitted. The incidence of failed and lost scrobbles has been fairly low for me personally (at approximately one or two a week), but it is definitely not a zero probability that your scrobbles may fail.

Even when scrobbles to Last.fm fail, the local database will always accurately record the ‘failed’ play, so the local database is always available as a ‘source of truth’.

If it is nevertheless important to you that Last.fm records absolutely everything listen to, you may want to take advantage of a service such as [The Universal Scrobber](#) (which I have no association with and therefore take no responsibility for!) to submit the occasional scrobble that fails to Last.fm by hand.

When you visit that website, you’ll need to log on with your Last.fm credentials and give the Universal Scrobber permission to post to your Last.fm account (much in the way described for giving Giocosos the same permissions in Section 8.2). Note that The Universal Scrobber site has no login of its own: it merely asks you to give it permission to use your Last.fm credentials (a permission which you can revoke at any time and never requires you to divulge your actual Last.fm username or password in any case).

In my case, for example, the play of Cherubini’s *Marche religieuse* failed one evening (and the Last.fm website was not contactable in a browser, either).

I was able to see the details of the ‘lost scrobble’ by running a report with this command:

```
giocosos --report=screen --dbname=main --reportdays=1
```

4000	2021-07-03	19:55	Percy Grainger	Four Settings from Songs of the North (Jones - 1998)	Vocal	00:07:39
4001	2021-07-03	20:01	Luigi Cherubini	Marche religieuse (Muti - 1984)	Choral	00:05:16
4002	2021-07-03	20:09	Arvo Pärt	Für Lennart in memoriam (Kaliuste - 2009)	Orchestral	00:07:22

...and therefore could fill in the Universal Scrobber submission form (accessed from the left-hand menu under the ‘Scrobble manually’ menu item) like so:

> Scrobble manually

Here you can manually enter your own track information to scrobble. Individual scrobbles will have the current timestamp.

Artist

Luigi Cherubini

Track

Marche religieuse (Muti - 1984)

Album

Optional

Album artist

Optional

Duration (in seconds)

316

Scrobble

Scrobble with options...

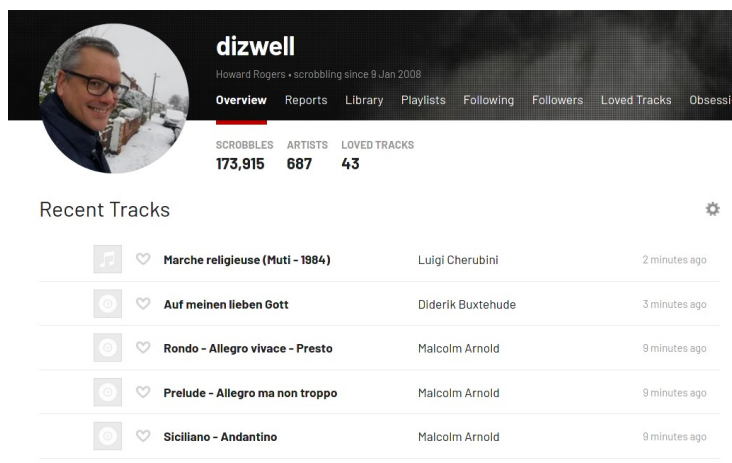
Clear

Technically, five pieces of information are requested, but only two are actually required -though I’ve chosen to submit the duration of the *Marche religieuse* anyway (measured in seconds, so a bit of mental arithmetic is required to convert Giocosos’s claim of ‘5 minutes 16 seconds’ into 316 seconds!)

Click the ‘Scrobble’ button when ready.

Not a lot will happen when you do, but a message should appear in a black ‘lozenge’ in the top right-hand corner saying ‘Success - The track has been scrobbled’. Don’t click ‘Scrobble’ too often thinking nothing has happened: you’ll end up submitting the same piece of music multiple times if you do!

Back in your Last.fm profile page, you should see the results of the new manual scrobble immediately (though you’ll need to refresh the page first, of course!):



The new track has been scrobbled successfully, though it doesn’t appear in the order in which Giocosos knows about. Here, Last.fm thinks it came after something by Buxtehude -which wasn’t listed in my original report from Giocosos at all. The absence of Buxtehude from the earlier Giocosos report is, however, merely a matter of timing. Here’s the same report produced a little later on:

3998	2021-07-03	19:29	Percy Grainger	Marching Song of Democracy (Rundell - 1997)	Orchestral	00:06:36
3999	2021-07-03	19:47	Francesco Geminiani	Concerto Grosso No. 4 (Manze - 1999)	Concerto	00:09:53
4000	2021-07-03	19:55	Percy Grainger	Four Settings from Songs of the North (Jones - 1998)	Vocal	00:07:39
4001	2021-07-03	20:01	Luigi Cherubini	Marche religieuse (Muti - 1984)	Choral	00:05:16
4002	2021-07-03	20:09	Arvo Pärt	Für Lennart in memoriam (Kaljuste - 2009)	Orchestral	00:07:22
4003	2021-07-03	20:16	Malcolm Arnold	Little Suite for Brass Band No. 1 (Howarth - 1993)	Orchestral	00:07:32
4004	2021-07-03	20:21	Diderik Buxtehude	Auf meinen lieben Gott BuxWV179 (Stella - 2011)	Keyboard	00:05:02
4005	2021-07-03	20:27	Witold Lutosławski	Overture for Strings (Wit - 1996)	Orchestral	00:05:19

So now you can at least see why Last.fm is reporting things by Arnold and Buxtehude... but clearly, my Cherubini play came *before* those as far as Giocosos is concerned, where Last.fm now thinks it came after.

Also note that Giocosos lists Arnold’s *Little Suite* as a single item -whereas it was scrobbled to Last.fm as multiple ‘tracks’. Remember that Giocosos plays (and records) whole compositions, though in deference to Last.fm’s primarily ‘track-based’ approach, it will always submit the four movements of a symphony to Last.fm as separate ‘tracks’, whilst recording only the play of the entire symphony as a single row in its own PLAYS table. Whether you choose to similarly manually submit ‘lost’ scrobbles to Last.fm on a per-track or a per-composition basis is entirely down to you and how much manual typing you want to do!

The point to make by way of summary is that Giocosos can most certainly ‘lose’ scrobbles because of network connectivity or website availability issues beyond its control -but you always have the option to manual submit anything that was ‘lost’ afterwards, if you feel the need to do so. (Generally, I don’t bother!)

11.3 Fonts

Being a text- and console-based music player, Giocosso depends heavily on your choice of terminal font for the effectiveness of its appearance. You may be entirely happy with your existing terminal font, and that's fine for so. However, a couple of suggestions for highly-readable monospaced fonts seems reasonable at this point!

My own choice of font, which you've seen in most of the screenshots in this document, is Century Schoolbook Mono BT, which is available for free download at best-font.com. The only downside to Century Schoolbook Mono BT, it seems to me, is that it can be difficult to distinguish capital Ohs from zeroes: the zero is not slashed or dotted, for example.

Anonymous Pro Minus fixes that particular issue and is freely downloadable from marksimonson.com. The big drawback of Anonymous for me is that it is a sans serif font (in other words, there are no little lines and strokes to 'finish off' the look of certain letters). That's not a problem for some, but I definitely prefer serifs!

So, for another excellent mono-spaced font that this time *does* do the serifs (and also makes Ohs and zeroes easily distinguishable), you could do a lot worse than Go Mono, available for nothing at fontlibrary.org.

11.4 Aliases

There is no doubt that there are a lot of run-time switches that can be used to alter or influence the work that Giocosso does -and having to remember to spell them out every time you want to play a bit of music can get very tedious very quickly!

For example, here's the command I've been using to run Giocosso since the beginning of July 2021:

```
giocosso --dbname=main --selections=99 --device=plughw:2,0 --pause=5  
--unplayedworks --maxduration=5 --scrobble --timebar=0
```

In plain English, that lot stands for 'run in Random Play Mode, play 99 pieces of music at a time and then quit; play to hardware device 2 (my external DAC), wait 5 seconds between plays of the separate choices of music; only select things to play which have not been played before, and make sure they are all less than five minutes long; once you've finished playing each piece, submit the 'play' to Last.fm; and the fact that you've just played some Mozart or some Vivaldi shouldn't be a bar to you playing some Mozart or Vivaldi immediately afterwards,

Phew!

Thank heavens for the up-arrow on the Linux command line, allowing you to recall previously-typed commands, because I'd be unlikely to be able to type that reliably more than once, I think!

But there's an alternative, too: Linux's ability to use 'aliases' to represent long combinations of commands with a short one. The specifics will probably depend on what distro you're running, but on Manjaro, I have a `$HOME/.bashrc` file in which aliases are defined. Here's some I prepared earlier!


```

GNU nano 5.7 .bashrc
alias np='nano -w PKGBUILD'
alias more=less

alias giocosoplay='giocosop --dbname=main --selections=99 --device=plughw:2,0 --pause=5 --unplayedworks --maxduration=5 --scrobble --timebar=0'
alias giocosoupdate='giocosop --checkver'
alias giocosorefresh='giocosop --refreshdb --dbname=main --musicdir=/sourcedata/music/classical'

alias splitflac='cao --asunder -x'
alias splitape='cao --asunder -x'
alias apetoflac='auac -i=ape -o=flac'
alias ripsacd='/usr/bin/sacd_extract -i 192.168.137.40:2002 -s -z -2 -o /home/hjr/Music'

```

You should be able to see that I've created three new 'giocosop commands', and described as being some combination of long giocosop options. Thus **giocosoupdate** is the 'short' command I'd use to get Giocosop to check for software updates, without having to remember that doing so actually requires the use of the `--checkver` parameter.

Similarly, I've said that the new command **giocosoplay** is equivalent to "giocosop --dbname=main --selections=99 --device=plughw... and so on and on". With that equivalence set in my `.bashrc` file, I can now just type 'giocosoplay' and all those myriad switches will be submitted automatically and without me having to remember any of them.

Commands which are aliased can still have extra parameters added. If, for example, I typed the command **giocosoplay --selections=3**, the first part of the command would mean 'run Giocosop with all those runtime parameters including the one for playing 99 selections, and then set selections equal to 3'. The extra parameter I supply gets bolted on to the end of all those I've said the basic 'giocosoplay' command should imply.

That's quite an interesting example, too, because `--selections=99` is actually part of the alias, so I'm essentially saying (abbreviating heavily!): `giocosop --selections=99 --selections=3`. When confronted with the same parameter twice in the same invocation, Giocosop always takes the last value as the 'truth'... so, my **giocosoplay --selections=3** command *will* actually trigger the playing of just three pieces of music before quitting, not the 99 implied by the alias.

Let's take another example. Say I issue this command:

```
giocosoplay --selections=3
```

...and this happens:

```

=====
giocosoupdate: giocosop --checkver  🐉 Giocosop: The Classical Music Player  🐉
giocosorefresh: giocosop --refreshdb -- Copyright © Howard Rogers 2021 /sda/music/classical'
                                     Version 1.00 - Randomised Play, using database MAIN
flac: 'cao --asunder -x'
=====
flac: 'auac -i=ape -o=flac'
=====
Exceeded number of allowed tries (50) finding a piece of music meeting all your selection criteria.
Re-run Giocosop with different selectors and filters.
Quitting in the meantime.
HJR@nano
=====

```


Well, that looks to be an error and therefore something's not right! Well: read the error text carefully. Giocosos says it can't find any music to play that meets my selection criteria -and, as it happens, this extract from a Giocosos Stats report will tell you it's entirely correct:

Duration Range	Total	Played	Unplayed
1 Less than 5 minutes:	1391	1391	0
2 Between 5 and 10 minutes:	2247	863	1384
3 Between 10 and 30 minutes:	13463	435	3028
4 Between 30 and 60 minutes:	4598	570	4028

That's showing you that of all my music that has a duration less than 5 minutes, none of it is unplayed. I've played every single piece of it at least once. Now check what my **giocosos-play** alias means again:

```
...--unplayedworks --maxduration=5...
```

So my alias implies 'only play things that are hitherto unplayed AND which last less than 5 minutes'... but the Stats report shows you there is no sub-five-minute music I haven't yet played!

But now let me issue this command:

```
giocosos-play --selections=3 --maxduration=10
```

So, I'm still saying "giocosos-play", which implies "unplayed, no longer than five minutes" but I've added another parameter which says "no longer than *ten* minutes" (and the earlier stats report shows you I've got plenty of under-ten-minutes music I haven't yet played).

Here's how Giocosos responds to that mixture of requests:

```

  卐 Giocosos: The Classical Music Player 卐
      Copyright © Howard Rogers 2021
  Version 1.00 - Randomised Play, using database MAIN

Now In:      /C/Camille Saint-Saëns/Orchestral/Caprice andalou (Dervaux - 1977)
Playing:     01 Track                               Duration:    00:09:26
Selection:   01 of 03 selections                     Time Bar:    Not in use
-----

Playing...
...works of previously unplayed recordings
...with a play-length of less than 10 minutes

Camille Saint-Saëns' Caprice andalou
Pierre Dervaux, New Philharmonia Orchestra

-----

Ending in: 00:09:20 - at 20:00:17 (approx.)

```

Well: it's working!

More particularly, the blue text in the middle of the screen tells you what selection criteria are actually in use: and it's the 'less than 10 minutes long' one. Because the `--maxduration=10` came *after* the `--maxduration=5` implied by the alias, it takes effect... and music can be found to play just fine in response.

You'll also notice that 'Selections' in the header area is displaying '01 of 03'... so that again tells you that the `--selections=3` manually bolted on to the alias has taken precedence over the `--selections=99` that's *implied* by the alias configuration.

In short: put your most common run-time options in an alias and you can still usually over-ride them manually when you need to, by *adding on* to the alias, rather than not using it entirely. To be honest, the alias I've shown you here isn't the best: sticking maxdurations and the unplayed filters in it are a bit foolish: those are things you might say occasionally as the mood takes you, not 'standard operating procedure' options! But I hope you get the general idea of what's capable with an alias, anyway.

Just remember; changes to your `.bashrc` file usually don't take effect until you either (a) log off and log back on again; or (b) open a new terminal session; or (c) type the command `". . / .bashrc"` in any existing terminal session in your `$HOME` folder.

To conclude: Linux makes it easy to create new 'commands' which imply or mean a command plus a whole bunch of run-time parameters. By typing in one short command, you're then really submitting a very long command. It's a very convenient feature and I accordingly recommend you make use of it as soon as possible in your use of Giocososo.

11.5 Transitioning from AMP to Giocososo

If Giocososo is your first venture in command-line, minimalist, optionally-scrobbling, optionally-randomising music players... then you can ignore this section! However, if you've been happily using AMP for the six months' of its existence and are now confronted with the problem of how you make the move from AMP to Giocososo, read on!!

Putting it as simply as I can: there's no real transition issue at all to worry about. Firstly, you can install Giocososo on a system that already has AMP installed and the two won't conflict in the slightest. The presence of Giocososo doesn't affect AMP at all, nor vice versa. You can even continue playing music with AMP whilst sometimes playing music with Giocososo.

In fact, there are really only two specific points about having both programs on the one PC to keep in mind:

1. The two programs must use different databases; and
2. The play history from AMP can be imported into Giocososo

The first point is significant in that it means music played in one program won't be recorded in the other. If I play Wagner's *Das Rheingold* in AMP one morning, Giocososo will not know that. Accordingly, Giocososo won't time-bar Wagner, nor will a report from *its* PLAYS table mention that bit of play history at all. The same is true the other way around: what is in Giocososo's database is completely unknown to AMP.

The two databases can therefore co-exist as independent entities: they will be stored in different physical locations (`$HOME/.local/share/amp` and `$HOME/.local/share/giocosso`, to be precise). That means the two databases can even have the same name as each other. The database name specified at the time of `--createdb` merely specifies the physical name the database file gets created with inside those two distinct folders. So there's no conflict and one database won't over-write the other.

Because of this 'product independence', though, the fact that you've configured AMP to scrobble doesn't mean Giocosso inherits that configuration or authorisation: so you'll need to configure Giocosso's scrobbling afresh, regardless.

The second of the two points I mentioned above has already been covered in [Section 9.7](#) of this manual: using the `--importplays=/location/of/amp/database/database-name.db` command, you can extract the existing AMP play history and pump it into the empty Giocosso database, thereby bring your play history across from one product to the other. It's a one-way journey, of course: AMP cannot import a Giocosso play history. It's also something you should only really do once, so the moment you've done it, you should stop using AMP to play music: any new plays done with it won't then be importable into Giocosso, short of creating a brand new Giocosso database and doing a fresh import there (which then means plays recorded in the first Giocosso database are lost).

For this reason, I'd recommend having both programs installed together. Continue to use both as the mood takes you until you are happy with the way Giocosso works (which shouldn't take too long, I think!). When you're finally ready, do a once-off export/import of PLAYS from AMP to Giocosso... and then never using AMP again. Uninstalling AMP at this point is simply a matter of issuing the command: `sudo rm -rf /usr/bin/amp.*`.

11.6 Tested Distros

In general, I tend to test my software out on whatever distros happen to be in [the top 10 Page Hit Ranking table at Distrowatch](#) at the time. Of course, distros move up and down the rankings over time, so what's in the top 10 today may not reflect what was in the top 10 when I was developing Giocosso! For the avoidance of doubt, therefore, here are the distros on which I've run and tested Giocosso and its desktop integration capabilities.

On the whole, Giocosso is likely to run fine on practically any distro -provided the software prerequisites are met, particularly since a lot of distros are built from other distros I've definitely tested. For example, I haven't tested on Zorin but I expect Giocosso to run on it just fine because Zorin itself is based on Ubuntu ...and I *have* tested on Ubuntu.

Meeting the software prerequisites can be a little tricky at times, though, because ImageMagick is installed on some distros by its mixed-case name, and others all in lower-case. Similarly, some distros use 'fd', some 'fd-find' and one uses 'fd-find' but calls it 'fd'! If your distro is not on the specifically-supported list, the generic instruction to install packages will still be shown -but you'll have to work out for yourself whether your particular distro knows a program by one name or another.

Unless the notes indicate otherwise, Giocosso worked fine on all distros listed! Remember that Giocosso integration can only work on Gnome, KDE and XFCE desktops at present.

Distro	Version(s)	Desktop Environment(s)	Distro Notes
Arch	'rolling'	GNOME	5
Solus	4.2	Budgie:GNOME	6
Ubuntu	20.04.2 LTS (Focal Fossa)	Ubuntu:GNOME	
EndeavourOS	2021.04.17	Budgie:GNOME	6, 7
Pop!_OS	20.04 LTS	pop:GNOME	8
openSUSE	15.3, 20210618	KDE	9
Linux Mint	20.1 (Ulyssa)	XFCE	
MX	10 (buster)	XFCE	
Raspbian	10 (buster)	LXDE	10
Debian	10 (buster)	MATE	11
Fedora	34 (Workstation Edition)	GNOME	12
Manjaro	'rolling', 21.0.7	KDE	13

General Notes:

1. All distros (except Raspbian and Manjaro) were run in KVM virtual machines with 8GB virtual RAM and 8 vCPUs. I *assume* physical installations onto real PCs and laptops are the same as virtualised ones, but cannot prove it!
2. Version is whatever is reported by `cat /etc/os-release | grep -i version`, unless that returns blank, in which case it's whatever `cat /etc/lsb-release` reports. Desktop environment is whatever is reported by `echo $XDG_CURRENT_DESKTOP`
3. All distros were installed with default options. The O/S was then immediately and fully updated. Giocosio was the first non-default program installed post-installation and post-update, following a reboot. Whatever software Giocosio prompted to install was then installed following Giocosio's suggested commands.
4. Some distros in the Distrowatch top 10 are derived from others (usually Arch or Ubuntu). If Giocosio has been tested on one of those parent distros, I generally haven't felt the need to test it specifically on the derived distros (Pop!_OS and MX being notable exceptions).

Distro-Specific Notes

5. Arch was built with a Gnome desktop, as per the instructions at absolutelybaching.com. As a 'rolling' release, it doesn't report a version number, just the word 'rolling'. There's no `/etc/lsb-release` on Arch by default, either... and when it's installed, it doesn't tell you anything other than 'rolling', too.

6. Budgie allegedly uses its own unique desktop environment -but it reports as essentially Gnome with knobs on! Gnome-equivalent integration, playback and album art display all work fine in consequence.
7. I chose to install EndeavourOS with the Budgie desktop, but MATE, Cinnamon, KDE, Gnome, XFCE and pretty much all others are possibilities. I did not test EndeavourOS with any of these other desktop environments, though -and MATE and Cinnamon definitely won't work with Giocoso's --integrate option. EndeavourOS doesn't report a version number when inspecting /etc/os-release. The 'BUILD_ID' is listed as 2021.04.17, however.
8. Pop!_OS installed by default with an X display numbered 1; everyone else numbers at 0. Code was modified to deal with this and all worked fine thereafter.
9. openSUSE was tested in both 'static' (i.e., 15.3) and 'rolling' (i.e., Tumbleweed) versions, though the KDE desktop was used for both. Static reports version 15.3, Tumbleweed reports what looks like a build-date (of 18th June 2021). Both types of openSUSE require the addition of the Packman repository to get ffmpeg (which Giocoso will explain how to do, if you haven't already added it).
10. My Raspberry Pi uses Raspbian, which is basically Debian built for ARM. Accordingly, version reports precisely as if it were Debian, but the desktop is the lightweight LXDE, with which no Giocoso desktop integration is, unfortunately, possible.
11. Debian was run with the MATE desktop: no integration testing is possible with this desktop environment. Everything else worked fine. Debian with KDE, XFCE or Gnome desktop environments can be expected to work perfectly (but not actually tested!)
12. According to `echo $XDG_SESSION_TYPE`, I was running 'wayland' (which has been the default display server protocol for Fedora since version 25, back in 2016). I expected trouble from Wayland, but there was none. Integration, playback and album art display all worked fine, first time of asking. Fedora doesn't ship ffmpeg in the standard repositories, so Giocoso-on-Fedora requires (and prompts for, if necessary) the installation of the RPM Fusion repository first.
13. Based on Arch, Manjaro reports 'rolling' as its version, but in `/etc/lsb-release`, it reports 21.0.7.

I should perhaps conclude by mentioning that Manjaro is the desktop I use every day; Ubuntu is installed on a physical laptop I use sometimes; and Fedora is on another laptop I use occasionally. Those three distros are the ones I'm most familiar with in real-world, physical PC-based situations. Over time, therefore, you may find that (say) Pop_OS! changes something that breaks Giocoso... and I wouldn't know about it for a long time, because I simply don't use it, except in a testing virtual machine.

The short version is: if you are using a 'niche' distro, I will do my best to support it in the future, but cannot guarantee anything. If you are on one of the more mainstream distros, I'm likely to spot and fix breakages more quickly.

Appendix A:

Differences between Giocosos and AMP

Giocosos is built on the same fundamental code-base as the earlier AMP (“Absolutely Batching Music Player”), but there are some significant differences between the two players which users migrating from AMP may want to familiarise themselves with, as follows:

- AMP scrobbled unless you tell it not to (assuming scrobbling had been configured). Giocosos doesn't scrobble unless you positively ask it to with the `--scrobble` parameter (or the `GIO_SCROBBLE=1` environment variable).
- AMP Scrobbling required the downloading and running of a completely separate `amp-scrobber.sh` shell script. Giocosos scrobbles ‘natively’ (without the need for extra scripts, etc).
- When running with `--report`, AMP exported the entire record of plays, every time. Giocosos can do full exports, too, but can also do differential ones: i.e., only export those plays which weren't on the previous export. It achieves this by keeping a record of the largest play number it ever reached to during an export.
- AMP had a lot of odd statistical selection overrides (i.e., `--levelup`, `--xlevelup` and `--xxlevelup`). Those have all been removed from Giocosos. The `--unplayed` and `--unplayedworks` parameters continue to exist, however, and it is now felt that they are able to boost under-performed composers and works sufficiently by themselves
- AMP had a fast database refresh mode (which was the default and the only thing which really populated the tables AMP selected from) and a full refresh, which went through every track in a music collection, populated a tracks table, but then didn't really do much with the data after that. Giocosos has dispensed with the fast refresh. It always does what AMP would have called a ‘full refresh’, and though this is much slower than the fast refresh, it allows it to collect information from music tracks (such as their length) which AMP was unaware of. One known drawback of this approach: adding one new CD to a collection requires a complete refresh to take place. One big advantage of this approach: selections using `--minduration` and/or `--maxduration` now work hundreds of times faster (see the next point for elaboration).
- AMP only applied `--min` and `--maxdurations` as filters after a music selection had been made. This meant having to iterate through a select-test for duration-reject loop, until something luckily passed the duration test and didn't need to be rejected. Giocosos now stores the duration of all recordings in its database, so duration filters are applied at the selection phase, eliminating the need to iterate through a test/reject cycle. The data is known to meet the duration criteria by the fact it got selected in the first place. Naturally, this makes using the duration filters much faster.
- AMP had a fairly convoluted two-mode method of working that was sometimes three modes! That is, you could run it directly in a folder, and it would play music it found there. Or you could run it in a folder which was a folder containing other folders that themselves then

contained music, and it would randomly pick a folder and play its music. Or you could run it with a `--dbname` switch and it would select something at random from the database. Giocosos simplifies this model and only has two modes of play: either with a database (in which case, what gets played is random) or by running it directly in a folder containing music, in which case that folder's music is played, without randomness. If you run Giocosos in a folder that is merely parents of other folders that contain FLACs, you will be told that no FLACs are present in the current folder, and Giocosos will simply quit.

- AMP could not play a specific recording via the database: if the database was used, the selection of what to play was *always* random (though you could affect the selection by asking for a specific composer, a specific genre and so on. Giocosos allows you to ask for specific recording to be played with the new `--record_number=x`, where `x` is the specific recording number as stored in the database's RECORDINGS table. If you know the recording number, you can therefore invoke Giocosos with a database and get it to play a specific recording, whereby its details are then subsequently added to the PLAYS table (and can be used in determining timebars and so on).
- A new `--albumlist` option allows Giocosos to generate and display a report of album titles and their album numbers, to help with the new 'play specific recording' feature itemised above. The report is ordered by composer name, so though it's likely to be rather long for large music collections, it should be quick to find a specific recording. Additionally, a `--composername=yyyy` filter can be added when generating the albumlist, and that will only output that composer's compositions, making finding a specific recording number even easier. This is all functionality that AMP lacked.
- Giocosos can import the PLAYS table from a previous AMP installation (using the `--importplays=xxxx` runtime parameter, where 'xxxx' is the full path to the old AMP database). You can therefore comfortably transition from AMP to Giocosos without losing precious data about what music you've been listening to for the previous many weeks or months. This functionality also allows Giocosos to re-import plays from another Giocosos database, so database recovery is now possible.
- AMP either used the default audio device, or you had to specify a specific device with a `--device` runtime parameter. Giocosos follows those rules, too, but can also use a `GIO_AUDIODEVICE=xxxx` environment variable to tell it how to play music. This is important to be able to do, because Giocosos can be invoked from within a file manager (such as Dolphin or Nautilus) with a right-click, at which point there is no opportunity to pass any device information. The use of the environment variable therefore allows this right-click-and-play-folder functionality to still use a user-specified audio device, in environments where the choice of device makes a significant quality difference.
- As just mentioned, Giocosos has been added to context-sensitive (i.e., right-click) menus for KDE's Dolphin file manager. AMP was never integrated into the OS like that.
- AMP used to require you to say `--usedb` and `--dbname=xxxx` before it would use a database to trigger randomised plays/. Giocosos takes a slightly more rational approach: if you've

mentioned a database name with the `--dbname` parameter, it's pretty obvious you intend to use it! Therefore, in Giocosio, `--dbname=xxxx` is the only parameter you need to supply to trigger database-sourced random playback: the `--usedb` parameter has been removed entirely.

- When using a database, AMP still liked you to specify `--musicdir=yyyy` so that it could strip the supplied parameter value from the full paths to music found in the database. Without it, AMP would work, but the display of what folder's music was being played would be very long. Giocosio now stores the `--musicdir` path supplied when creating/rescanning a database, so there is no longer a need to mention it when using the database for playback. It already knows the 'root' of your music collection and automatically knows to strip that from the full paths of music folders for display purposes.
- Album art is still displayed by default and the `--artsize` parameter still takes values of small, medium or large. However, it now also takes a value of **none**, which is how you stop album art being displayed at all. AMP could also switch off album art display, but used a separate parameter called `--noart` to do it. That parameter no longer exists in Giocosio.
- Still on album art: Giocosio now adds the ALBUM tag text to the bottom of the album art display, so that just by looking at the art work (which might have been on a CD containing a dozen different pieces) you can tell precisely what composition is being played.
- The GPL v.2 License can now be viewed within Giocosio (using the `--license` runtime parameter), instead of having to visit a URL and download it manually, as in AMP days.
- Giocosio has an option to `--register` the product (basically, email me something so I know how many people are using it and on what distros ...see **Appendix C** for details)

For the most part, despite all these listed differences, if you were comfortable with getting AMP to play your music collection, you'll have no difficulty transitioning to Giocosio. Basically, just remember to actively choose scrobbling (if you want it at all) and remember that `--usedb` is now redundant: the rest is all just details ☺

Appendix B:

The Requirement for Sudo Privileges

Giocosos does **not** require sudo (i.e., acquired root privileges) for ordinary operation, but you will need it for (a) installation and (b) desktop environment integration (that is, if you're going to use the `--integrate` option).

Sudo privileges mean you stay you, and you supply your password, but you acquire temporary rights to run with root privilege and are thus allowed to create or modify files in restricted folders (like `/usr/bin`). It's a safer way of doing things than *becoming* root directly, which would give you permission to do anything at all for as long as you liked.

Most distros, during installation, tend to ask you for a user account and *automatically* give that initial user account sudo privileges, so you do not then need to do anything at all. Giocosos or related scripts will prompt you for your 'sudo password' if necessary, and you just supply your ordinary user account password at that point, and all will be well.

But not all distros do this: Debian, for example, does not and trying to run a sudo command produces this foreboding error:

```
hjr@debian:~$ sudo ls /usr/bin
hjr is not in the sudoers file. This incident will be reported.
hjr@debian:~$
```

...which is quite funny on a home PC where the only person to report to is yourself!

Anyway, for distros where you cannot automatically exercise sudo rights, here's the workaround. First, become root. Second, set the default editor to **nano** or even a graphical text editor of your choice: if you need help setting sudo, you do not want the pain and hassle involved with using most distros' default text editor, which is **vi**. Finally, you invoke the **visudo** program and add one line of text to the existing file and save it.

It's just four steps, in other words. Here are the first three, spelled out, as you'd type them in a terminal:

```
su - root
export EDITOR=nano
visudo
```

The first command requires you to be able to provide *root's* password when asked. Assuming you know that, you're in business. When the visudo program runs, you want to find the line which reads:

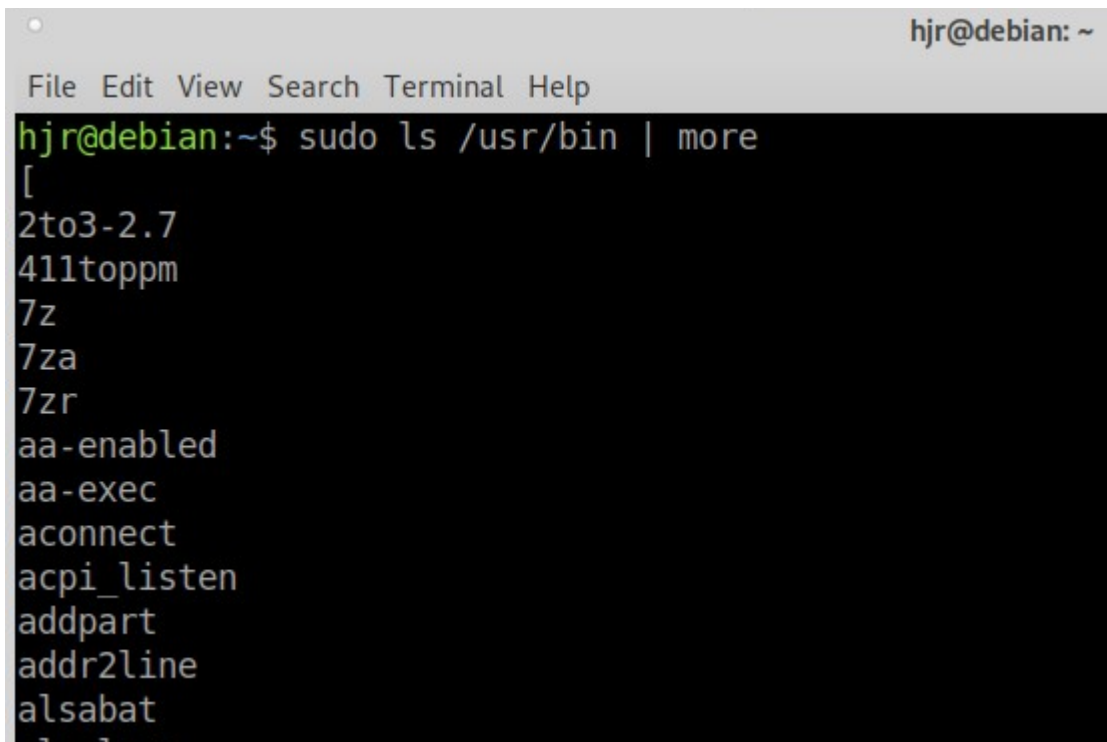
```
root    ALL=(ALL:ALL) ALL
```

... and type, **underneath it**, an equivalent line such as:

```
hjr ALL=(ALL:ALL) ALL
```

The first part of that new line is your own username: in my case, it's 'hjr', but you replace that with whatever username you *actually* log onto your Linux PC with.

Then you save the modified file: in nano, you press Ctrl+X, then tap 'y' to confirm the save, and press Enter to save the modified file with the filename proposed. Job done: you can now stop being root (so, type `exit`). And now you can run any command with root privileges, just by prefixing the command with the word 'sudo':



```
hjr@debian: ~  
File Edit View Search Terminal Help  
hjr@debian:~$ sudo ls /usr/bin | more  
[  
2to3-2.7  
4l1toppm  
7z  
7za  
7zr  
aa-enabled  
aa-exec  
aconnect  
acpi_listen  
addpart  
addr2line  
alsabat  
...
```

This time, the 'ls /usr/bin' command produces no error at all.

As I say, there are only two occasions when Giocosos requires the use of sudo: installation and integration with your choice of desktop environment.. and both of those occasions are strictly 'one-off' affairs.

You definitely do NOT need sudo privileges to run Giocosos for the purposes of ordinarily playing music!

Appendix C: Product Registration

Another run-time parameter is available to Giocosos users: `--register`. It doesn't really do anything much except display the following wall of text:

```
-----
      🎵 Giocosos: The Classical Music Player 🎵
      Copyright © Howard Rogers 2021
      Version 1.00 - Non-Play Activities
-----
```

Registration is entirely optional, but it would help me to know how many people are using Giocosos and what distros they're running it on. If you wouldn't mind emailing this code:

```
Manjaro Linux-L21.0.7-h7gdJSHGF....hcIZSps8ui9z0
```

...to: registration@absolutelybaching.com, it would be much appreciated and will also help me improve the program in the future. The code is a randomly-unique string that means nothing and divulges no personal information, prefixed by the identifier/version number of the Linux distro you're using.

No aspect of Giocosos's performance or functionality is impacted if you choose not to send it. Many thanks, Howard Rogers.

Basically, the option generates a unique, random and completely meaningless 24-character code and then prefixes it with the name of your distro (as contained in the `/etc/os-release` file) and the version number of that distro, either contained in either the `/etc/os-release` file or the `/etc/lsb-release` one. If neither file contains a version number, it's reported as '0'.

The random string element of the registration code is derived from `/dev/urandom`, so it has no significance or meaning and does not and cannot be used to identify anything about you personally or your PC. It's uniqueness simply means I can tell one person's Fedora 34 apart from another one's.

Registering consists of sending this 'code' in full to registration@absolutelybaching.com. You do that entirely manually using whatever email client you like. I'd suggest using the registration code as the email's subject, leaving the body empty; but if you stick it in the body of the email, that's fine too.

The process is entirely optional and if you choose not to register Giocosos in this way, it's functionality is not altered or restricted in any way. You also gain no specific benefit or advantage by registering.

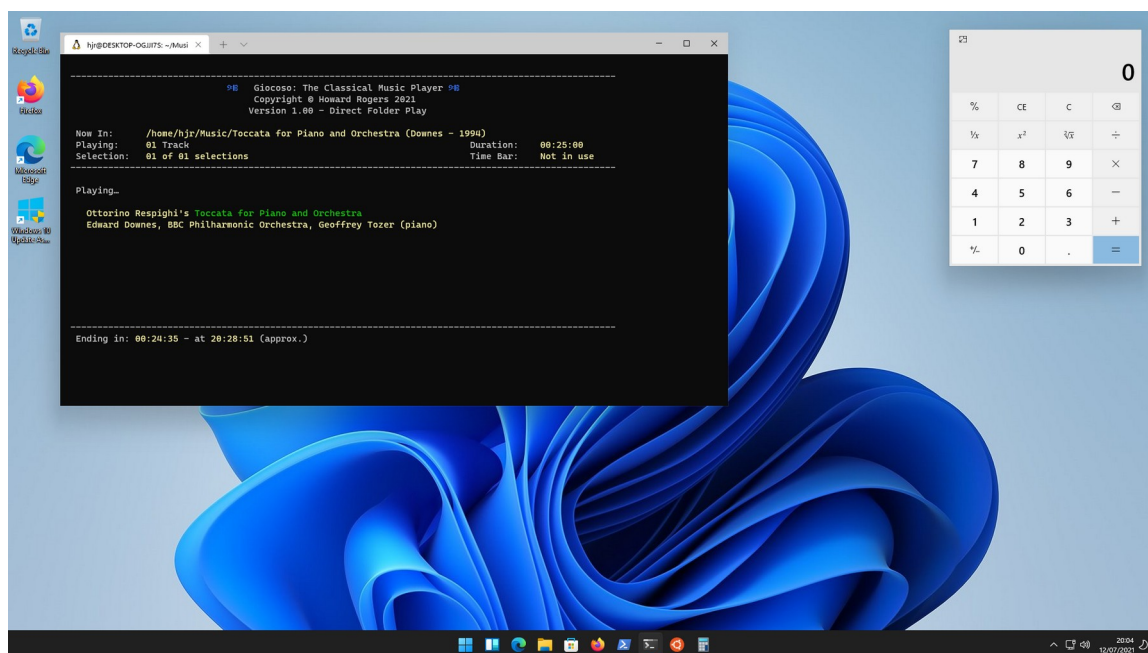
Registering Giocosos, though, lets me know a little about how many people are using it and on what distros. I'll test more on distros I know to be in use than on those I know no-one is using, for example.

So: registering the product by emailing me a unique code is helpful to my future development efforts. But if you prefer not to, that's entirely fine, too. This complete optionality is why I didn't document the feature back in **Section 10**, along with all the other run-time parameters. I want you to feel comfortable registering it or not, as you deem appropriate -but I nevertheless hope you *will* send me that email after all!

Appendix D: Giocososo on Windows

Way back on page 6, in the very first sentence of this manual, I pointed out that Giocososo is a *Linux Music Player*. It doesn't run on FreeBSD or Solaris, for example (though it might do: I just haven't tested it to find out!). And it *definitely* doesn't run on Windows.

Or does it?!



Well, clearly it *can*... but it is tediously difficult to make it do so right now (future versions of Windows 10 and 11 might make it considerably easier, though).

It's really beyond the scope of a user manual to explain in detail how this particular bit of wizardry was achieved, but in summary:

- As a Windows Insider, I was able to upgrade to Windows 11 and install the new Windows Subsystem for Linux version 2 with graphics (WSL2g)
- I then installed Ubuntu 20.04 from the Windows Store
- Quite a bit of fiddling with package installation and user permissions ensued!
- Giocososo installation and subsequent configuration and running then just happens as it would normally on a 'proper' Linux PC.

It's a lot of effort to go to, and frankly, I can't say I think it's an effort most regular Windows users are going to be brave enough to expend! As I say, if Windows 10 or 11 makes running WSLg easier in the future, my opinion on that might change.

Anyway, for now: I just wanted you to know that it *is* possible to run Giocososo on Windows, provided you run it on Linux-within-Windows-with-Graphics, via WSL2g.

Acknowledgements

I'd like to thank the developers of ffmpeg, ImageMagick, sqlite and all the other open source software that Giocosos uses and coordinates to pull off its music playing tricks: Giocosos is really only an 'orchestrating script', rather than a complex piece of software in its own right, and would be nothing without the real programming efforts of all those developers.

Giocosos's scrobbling abilities arise directly out of the original capabilities of the [MOC-Scrobbler](#), written by a user called 'pachanka'. I have hacked the original work about quite a bit to enable a couple of its core capabilities to be embedded directly inside Giocosos... but, fundamentally, without all that work by someone else on the MOC Scrobbler, Giocosos couldn't scrobble at all. So, I'm grateful to pachanka -with whom I've had no communication about what I've done to his or her work, incidentally, so any coding horrors arising out of my tinkering with the original work are entirely my own fault.