LCIO Command Line Tool

LCIO Command Line Tool

The LCIO command line tool provides a number of useful utilities from the command line.

Installation

To install the runnable LCIO jar, execute the following from the LCIO root directory.

```
mvn
```

This will create a standalone, runnable jar such as:

```
target/lcio-2.4.4-SNAPSHOT-bin.jar
```

The exact file name will depend on the LCIO version.

Overview

The command line tool is launched using the *java* command, with the following syntax.

```
java -jar ./target/lcio-[VERSION]-bin.jar [command] [arguments]
```

For instance, this is the command that will print out the usage instructions on my system.

```
java -jar target/lcio-2.4.4-SNAPSHOT-bin.jar
```

With no arguments, the usage statement is printed.

```
usage: LcioCommandLineTool
Commands:
compare
concat
validate
siodump
print
stdhep
split
random
count
merge
 -h    Print lcio command-line tool usage.
 -v    Set the verbosity.
```

Similar to the *cvs* command, the LCIO command line tool takes the name of command as its first argument. The command should be one of the following.

- **compare** - Perform a diff on two LCIO files, reporting the differences.
- **concat** - Concatenate two or more LCIO files together.
- **validate** - Perform some very simple validation of an LCIO file.
- **siodump** - Print low-level LCIO information using the SIO interface.
- **print** - Format and print all the collections and metadata of an LCEvent.
- **stdhep** - Convert StdHep events into LCIO MCParticle collection.
- **split** - Split an LCIO file into smaller chunks.
- **random** - Generate a random LCIO event. (e.g. for testing purposes)
- **count** - Count event and run headers of 1 or more LCIO files and report statistics.
- **merge** - Merge 2 or more LCIO files together.

Commands

Compare Command

The *compare* command takes 2 or more input LCIO files and compares them to find differences.

```
usage: compare
 -f    Add a file to compare (Must have at least 2).
 -n    Set number of events to compare.
 -s    Set number of events to skip.
 -v    Set verbosity level (must be between 1 and 4).
```

For instance, this command will compare the second events of files *file1* and *file2*, with high verbosity.

```
lcio compare -f file1.slcio -f file2.slcio -n 1 -s 1 -v 4
```

Extra arguments will be interpretted as additional input files, so the above could be written this way instead.

```
lcio compare -n 1 -s 1 -v 4 file1.slcio file2.slcio
```

The comparison is quite detailed, so LCIO files will not be considered identical unless they are basically exact copies of each other.

Concat Command

The *concat* command takes 2 or more input LCIO files and concatenates them together into a single output file.

```
usage: concat
 -f    List of input files, 1 per line.
 -i    Add an input file.
 -o    Set the name of the output file.
```

The following will concatenate together *file1* and *file2* into *outputfile*.

```
lcio concat -f file1.slcio -f file2.slcio -o outputfile
```

The *concat* command also accepts lists of files, with one file per line.

```
file1
file2
...
```

Supposing that the above was contained in a file called *flist.txt*, the *concat* command can read this file list and concatenate together all the files in the list.

```
lcio concat -i flist.txt -o outputfile
```

Validate Command

The *validate* command takes 1 or more LCIO files as input and performs two simple checks on them. Firstly, a simple read test is done. If the read test fails, the LCIO file is considered invalid. Secondly, a comparison is performed between the major and minor LCIO version numbers in the file and from command line arguments. Any version mismatches are printed to the terminal.

```
usage: validate
 -M    Set major version.
 -f    Add an LCIO file to validate.
 -m    Set minor version.
 -n    Set number of events to read.
```

This command will perform one read test and check if the files were generated with the LCIO 1.5 version.

```
lcio validate -f file1.slcio -M 1 -m 5 -n 1
```

As in other commands, extra arguments are interpretted as paths to additional input files, so the "-f" switch can be dropped.

```
lcio validate file1.slcio -M 1 -m 5 -n 1
```

Siodump Command

The *siodump* command prints out detailed information about the SIO blocks in LCIO files using the SIODump utility (written by Tony Johnson).

```
usage: siodump
 -f    Set the input LCIO file.
 -n    Set number of events to dump.  (Must be > 0) .
 -s    Set number of events to skip.  (Must be >= 0) .
 -v    Set verbosity level.  (Must be between 1 and 4) .
```

The following command will read the file *file1*, skip 1 event, and dump 2 events, with low verbosity.

```
lcio siodump -f file1.slcio -n 2 -s 1 -v 1
```

This command accepts only a single input LCIO file, so extra arguments are ignored, not treated as additional input files.

Print Command

The *print* command prints detailed information about LCEvents, including tables of records for each LCCollection of LCObjects. This command is a rewrite of the C++ dumpevent utility (written by Frank Gaede).

```
usage: print
 -f    Set the LCIO file to dump.
 -m    Set the maximum number of records to print per collection.
 -n    Set the maximum number of events to dump.
 -s    Set the number of events to skip.
```

The following command will dump 2 events from the file *file1*, skipping the first event and limiting the number of objects printed from collections to 100.

```
lcio print -f file1.slcio -m 100 -n 2 -s 1
```

This command accepts a single file. If there is no "-f" argument, the first extra argument is inpretted as the input LCIO file.

StdHep Command

The *stdhep* command reads in a StdHep file and converts the HEPEVT blocks into LCCollections of MCParticles according to an algorithm by Ron Cassell (Java implementation by Tony Johnson).

```
usage: stdhep
 -i    Input Stdhep file.
 -o    Output LCIO file.
```

This command converts *events.stdhep* into *lcio_events.slcio*.

```
lcio stdhep -i events.stdhep -o lcio_events.slcio
```

This command ignores extra arguments.

Split Command

The *split* command takes a single LCIO file and splits it into smaller chunks.

```
usage: split
 -i    The input LCIO file.
 -n    The number of events to split.
```

The following command will split the file *file1.slcio* into multiple files, each with a maximum of 5 events.

```
lcio split -i file1.slcio -n 5
```

The files are named using the scheme *ORIGINALNAME-FILENUM-EVENTNUM*, so the first file created by the above command would be called *file1-0-5.slcio*.

This command ignores extra arguments.

Random Command

The *random* command generates LCIO events filled with random data (which is basically nonsensical!). These LCIO files can be used for testing purposes.

```
usage: random
 -h    Print random usage.
 -m    Set the maximum number of objects in a collection.
 -n    Set the number of random events to generate.
 -o    Set the LCIO output file name.
```

The following command will generate 5 random events to the output file *events.slcio*, with maximum of 100 objects per LCCollection.

```
lcio random -m 100 -n 5 -o events.slcio
```

If the "-o" argument is omitted, the default output file name of "random_events" is used.

Count Command

The *count* command counts the number of event and/or run headers found in the input LCIO files.

```
usage: count
 -e    Print number of event headers.
 -f    Add an input LCIO file.
 -h    Print count usage.
 -r    Print number of run headers.
 -t    Print totals of all files processed.
```

The following command will print the number of run and event headers found in the files *file1.slcio* and *file2.slcio*, plus the total numbers overall.

```
lcio count -f file1.slcio -f file2.slcio -e -r -t
```

Extra command line arguments are interpretted as additional LCIO input files.

Merge Command

The *merge* command overlays events from different LCIO files into one output file. It is a rather complicated command with a lot of possible options.

```
usage: LCIO Merge command
merge command
 -i    Set input file list with format:
       [file_name],[n_reads_per_event],[start_time],[delta_time]
       This option is not usable with the -f argument.
 -T    Set the starting time (ns).
 -e    Set number of events to merge in from each input file per merged
       event.  (Default is 1)
 -f    Add an input file.
 -h    Print merge usage.
 -n    Set maximum number of output events.
 -o    Set the output file.
 -t    Set delta time (ns).
```

In its simplest form, this command accepts a number of input files, reads one event from each input file, and writes the resulting overlayed event into an output file. For instance, to merge together *file1.slcio* and *file2.slcio* into *file3.slcio*, execute this command.

```
lcio merge -f file1.slcio -f file2.slcio -o file3.slcio
```

The more complicated form of this command accepts a list of input files plus parameters associated with each one, including the number of reads to perform per output event, the starting time, and the delta time that is added for each overlay when there are multiple reads of the same file for one output event.

The command above could be replicated as follows if the file *flist.txt* contains the following lines.

```
/path/to/file1.slcio,1,0.0,0.0
/path/to/file2.slcio,1,0.0,0.0
```

The merge command then references this list.

```
lcio merge -i flist.txt
```

A delta time can be applied for simulating pileup, as in the following. This will pileup 5 events from the same input file, incrementing all times in the data objects by 5 nanoseconds per event.

```
/path/to/events.slcio,1,0.0,5.0
```

Backgrounds can also be overlayed using this command. One event will be read from *events.slcio* for each merged event, while 5 events will be read from *backgrounds.slcio* with a 5 nanosecond time increment for each background event.

```
/path/to/events.slcio,1,0.0,0.0
/path/to/backgrounds.slcio,5,0.0,5.0
```

In general, any arguments in the file list will override those from the command line.

Select Command

```
usage: select
 -f    Set the input LCIO file.
 -h    Print select usage.
 -i    Add a pattern to include.
 -o    Set the output LCIO file.
 -t    Add an LCIO type to include.
```

The select command extracts collections from an LCIO file and writes them out to a new file. Selection is performed either by a regular expression match on the collection name or by the type of objects in the collection.

The type is compared to the value of LCCollection.getTypeName() for each collection.

The patterns should be valid regular expressions, and, to be safe, should include the ^ and $ characters for marking the beginning and end of the expression.

It is possible to give multiple patterns and types. The collection need only match a single pattern or type. When both types and patterns are defined, the collection must match at least one type and one pattern to be included in the output file.

Do not include a pattern when only type selection is required. This will only do type matching and includes all collection names as matches by default.

This command selects all SimCalorimeterHits from an event file.

```
lcio select -f events.slcio -o calhits.slcio -t SimCalorimeterHit
```

This command selects all collections starting with "Hcal".

```
lcio select -f events.slcio -o hcal.slcio -i Hcal.*
```