IPOL Software Guidelines — version 1.00

2011-12-20

IPOL reviews, uses, publishes and distributes some software provided by the authors. With the requirements and recommendations expressed in these guidelines, we intend to facilitate the production and review of verifiable and usable software for reproducible research.

Contents

1 In Brief: Check List, Check Service and Examples

The list hereafter is a summary of the guidelines, to quickly check an IPOL program. Some are requirements, others are only recommendations. The guidelines are detailed and explained later in this document.

- zip or tar/gzip archive `name_version.{zip,tar.gz,tgz}`, less than 2 MB
- everything into a `name_version/` folder
- file names with `a-z`,`A-Z`,`0-9`,`-`,_,`.`
- no hidden file, backup or useless file, no binary
- C89, C99 or C++98 code tested with `gcc -std=xxx -Wall -Wextra -Werror`
- portable code, 32/64-bits, nothing specific to an operating system

- only `libtiff`, `libjpeg`, `libpng`, `zlib`, `fftw`, `cblas` and `clapack` external libraries
- compilation with `make` or `cmake`, only standard options, `make` uses `$(CC)` or `$(CXX)`
- command-line non-interactive interface
- max 1 GB memory, max 30 s computation in the demo environment
- can read/write in PNG, TIFF, PNM, EPS, SVG, VRML or PLY format
- copyright attribution and GPL/BSD license info in every source file
- patent warning if needed
- `README.txt` essential information
- correct, clean code in English
- max 80 characters per line, max 1000 lines per file
- `main()`, algorithmic and auxiliary code in different files
- detailed comments for every function and every implementation step
- example input data and result

A service to check an IPOL program against some of these guidelines is available with examples of programs following the guidelines at `http://tools.ipol.im/pkg/`. This service can be used by IPOL authors to verify their code before submission, and by reviewers as a preliminary validation of the software[1].

2 About this Document

2.1 Status

This document is the official IPOL software guidelines, version 1.00, published on December 20th, 2011. It is immediately applicable and obsoletes previous versions. The reference version is available on line at `http://tools.ipol.im/wiki/ref/software_guidelines/`.

[1] The absence of error reported by this service doesn't imply that all the guidelines are correctly followed. Some guidelines need a human review.

2.2 Revisions

When needed, future versions of this document will be published and will replace the current version. The current version will be kept and a summary of the differences will be provided. This revision will be announced on the IPOL website[2] and the IPOL discussion list[3].

2.3 Vocabulary

In this document, the term "IPOL program" is used to designate the reference program implementation of an algorithm submitted for publication in an IPOL article. An IPOL article **may** publish more than one program, an IPOL demo **may** use more than one program.

In this document, the words **must**, **must not**, **should**, **should not**, **recommended**, and **may** are used to express required, recommended, and optional items. Their interpretation is described in IETF RFC2119[4] and detailed in the context of these guidelines in Annex A.

3 Guidelines

3.1 1. Packaging and Content

3.1.1 1.1. Compressed Archive

An IPOL program **must** be packaged as a compressed archive file. This file archive can either be a single volume .ZIP compressed archive or a GZIP compressed tar archive[5]. The size of the compressed archive file **should** be less than 2 MB. In the remainder of this document, we will use the terms "zip archive" and "tar/gzip archive" for convenience.

Annex B of this document provides some examples of programs that can be used to produce such compressed archives.

[2] `http://www.ipol.im/`

[3] `http://tools.ipol.im/mailman/listinfo/discuss`

[4] `http://tools.ietf.org/html/rfc2119`

[5] These file formats are defined by the PKZIP APPNOTE specification[6], version 6.3.2, for the .ZIP compressed archive format, the IETF RFC1952[7] for the GZIP compressed format, and the POSIX.1 ustar definition[8] for the tar archive format.

3.1.2 1.2. Archive Name, Program Name and Version

The compressed archive file of an IPOL program **must** be named according to the `name_version.extens` pattern, where:

- `name` and `version` **must** consist only of lower case letters (`a-z`), digits (`0-9`), minus (`-`) and period (`.`) signs;
- `name` **must** be at least two characters long and start with a letter; it **must** indicate the name of the program; this name can be the name of the executable program file, or another name, at the author's will;
- `version` **must** start with a digit; it **must** indicate a version number for the program, in the sense that two different releases of the program **must** have two different version numbers; if no version numbering scheme is established for the program, the `YYYYMMDD` pattern based on the year, month and day of the release date **may** be used;
- `extension` **must** be `zip` for zip archives and `tar.gz` or `tgz` for tar/gzip archives.

3.1.3 1.3. File and Folder Names

All the files and folders extracted from the compressed archive **must** be located inside a base folder named `name_version`, where `name` and `version` are identical to those used for the compressed archive file name. Absolute path **must not** be used for files and folders extracted from the archive.

The name of all files and folders composing the IPOL program **must** consist only of lower or upper case letters (`a-z`, `A-Z`), digits (`0-9`), minus (`-`), underscore (`_`) and period (`.`) signs. They **should** start with a letter.

The names **should** provide a meaningful hint of the content of these files and folders.

3.1.4 1.4. Hidden and Useless Files

An IPOL program **should not** include hidden files or folders or by-products of the tools used by the authors, such as (but not limited to):

- files inserted by file managers (`.DS_Store`, `.directory`);
- folders inserted by version control managers (`.svn`, `.git`);
- backup versions (`filename~`, `filename.bak`).

The program **should not** be distributed with files not useful to build, use or study the implementation of the algorithm published in IPOL.

3.2 2. Implementation

3.2.1 2.1. Source Code

An IPOL program **must** include all the material necessary to build one or more executable program files implementing the algorithm published in IPOL. This material **must** be provided in human-readable source code form. An IPOL program **must not** be distributed with binary precompiled files if these files can be obtained from source code[9].

Annex C provides some information for IPOL authors to help them perform various frequent implementation tasks.

3.2.2 2.2. Programming Language

The source code of an IPOL program **must** follow the published standard syntax of one or more compiled programming languages. IPOL can currently only process C89 (ANSI C), C99, and C++98 (ISO C++)[10]. If the authors want to publish their program with another well-known and standardized compiled language (such as Fortran 90), they should contact the editorial board to investigate the possibilities.

IPOL authors **should** test their C and C++ source code with the `gcc` compiler in strict compilation mode[14] before submitting it to IPOL.

The source code **may** use the OpenMP 3.0[15] API for shared multiprocessing programming (parallel programming) but it **must** also compile and provide the same results (albeit slower) without OpenMP. Usage of OpenMP **must not** be tied to a specific number of processors and **must** only rely on the OpenMP standard, not on any vendor implementation.

3.2.3 2.3. Portability

The source code of an IPOL program **must not** require any extension of the language or its standard library, or any resource specific to a hardware environment, operating system or compiler. These extensions and resources **may** be used to achieve better performances if they are available but their availability **must** be detected during the compilation or execution and an alternative portable implementation **must** be used in their absence. This includes (but is not limited to)

[9] If the authors want to distribute binary versions, they can do it in IPOL but not via the compressed archive of the IPOL program.

[10] C89 is defined by the ANSI X3.159-1989 Programming Language C standard[11], C99 is defined by the ISO/IEC 9899:1999 Programming languages — C standard[12], and C++98 is defined by the ISO/IEC 14882:1998 Programming languages — C++ standard[13].

[14] C89, C99 and C++98 code can be tested with `gcc -std=xxx -Wall -Wextra -Werror` where xxx is `c89`, `c99` or `c++98`.

[15] `http://www.openmp.org/mp-documents/spec30.pdf`

- language dialects specific to a compiler (GNU C, Microsoft C);
- standard library functions specific to an implementation (`drand48()`);
- assembler code or "intrinsic functions" mapped to a processor instruction (`mulps`, `__builtin_ia32_mulps()`);
- operating system calls (Win32 `GetSystemTime()`, POSIX `gettimeofday()`);
- file system locations (`C:\Documents and Settings`, `/tmp`);
- code specific to a memory model environment (32-bits, 64-bits).

Special attention will be given to the Linux 64-bit environment because it is currently the primary environment for IPOL demonstrations. But the program **must** also be usable in other environments and this portability **must not** be limited to the Win32/POSIX alternative.

3.2.4 2.4. Dependencies

An IPOL program **must** not use external software components except for the libraries and APIs listed hereafter. The program **may** expect these software components to be correctly installed and configured during the compilation and execution:

- `libtiff`[16] version 3.x and `libpng`[17] version 1.4.x to read and write files, with their dependencies `libjpeg`[18] version 8.x, `zlib`[19] version 1.2.x;
- `fftw`[20] version 3.x (single and double precision) for Fourier transforms;
- `cblas`[21] and `clapack`[22] for linear algebra.

Other libraries can be examined and may be added to this list on request, if they are portable, widely used, with a stable API.

This restriction only applies to software components used by the IPOL program but not distributed in source form with the program. Annex C has more details about how some external code, including external library code, can be used in the IPOL program.

[16] `http://www.remotesensing.org/libtiff/`
[17] `http://libpng.org/pub/png/libpng.html`
[18] `http://www.ijg.org/`
[19] `http://zlib.net/`
[20] `http://www.fftw.org/`
[21] `http://www.netlib.org/blas/`
[22] `http://www.netlib.org/lapack/`

3.2.5 2.5. Compilation

An IPOL program **must** be compiled by an automated non-interactive build procedure with `make` or `cmake`. This build tool **must not** be configured to use any special compiler.

For example, `make` **must not** call `gcc` or `g++` but **must** use the `$(CC)` and `$(CXX)` variables instead. The default build procedure **must** use standard compiler options only[23] : `-c`, `-D`, `-E`, `-I`, `-L`, `-l`, `-O`, `-o` and `-U`.

3.2.6 2.6. Usage and Input/Output

An IPOL program **should** be minimal and only perform the algorithm published in IPOL. It **must** be usable from the command line environment without any user interaction, taking all its parameters from the command line.

An IPOL program **must** be able to read the input data and write the final output data in at least one of these formats[25]: PNG, TIFF or PNM for raster images, EPS or SVG for vector images, VRML or PLY for meshes, and plain text for other data. Other file formats can be added to this list on request, if they are clearly defined and widely used.

Annex C provides some information for IPOL authors to help them use external libraries to read and write images.

3.2.7 2.7. Computing Resources

In the demo environment, an IPOL program **should not** use more than 1 GB of memory and **must not** use more than 8 MB of stack memory space (for recursion, local variables and variable-length arrays). The program **must not** need more than 30 seconds to process typical data. For slow algorithms, this limit **may** be achieved with parallel processing or a limit on input size.

Annex C provides some information for IPOL authors to help them improve the performance of their implementation.

[23] Standard compiler options are defined by the POSIX c99 specification[24]. Other options depending on the environment (hardware, operating system, compiler) **may** also be provided, for example for an optimized compilation, but they **must not** be used in the default build procedure.

[25] PNG is defined by the IETF RFC2083[26], TIFF is defined by the Adobe TIFF 6.0 Specification[27], PNM (PBM, PGM and PPM) is defined by the netpbm documentation[28], EPS is defined by the Adobe Encapsulated PostScript 3.0 Specification[29], SVG is defined by the W3C Scalable Vector Graphics 1.0 Specification[30], and VRML is defined by the ISO/IEC Virtual Reality Modeling Language Specification[31]. There is no formal published specification of PLY, but this simple format introduced by the Standford 3D Scanning Repository[32] is documented on Paul Bourke's site[33]. Plain text output **should** be understandable by a human reader and easy to parse with a software.

3.3 3. Copyright, License and Patents

3.3.1 3.1. Copyright Attribution

Every source code file in an IPOL program **must** mention its authors in a copyright attribution line at the top of the file. This mention **may** be omitted in very simple files such as header code.

Every person whose contribution to this file is not trivial and implies some creative work **must** be credited. Of course, if the authors use or modify a file previously written by other persons, the copyright attribution to the previous authors **must not** be removed. The copyright attribution **must** include the years of production of the work, the full name and an e-mail address for contributor. It **may** also include other relevant information such as the employer, affiliation or web site.

An simple example for a single author can be:

```
Copyright (C) 2011, Jane Doe <jane.doe@example.org>
```

A complex example with many contributors can be:

```
Copyright (C) 1998-2003, Taro Yamada <taro.yamada@example.jp>
Copyright (C) 2005-2011, Juan Perez <juan.perez@example.es>
Copyright (C) 2011,      Marie Untel, ENS Cachan
                         <marie.untel@ens-cachan.fr>
```

3.3.2 3.2. Patent Warning

When the authors are aware or suspect that a source code file implements an algorithm which might be linked to a patent (the main algorithm published on IPOL or another algorithm used for this implementation), a patent warning **must** be inserted after the copyright attribution, in every file potentially linked to this patent. This wording is **recommended**:

```
This file implements an algorithm possibly linked to the patent
<REFERENCE OF THE PATENT>.
This file is made available for the exclusive aim of serving as
scientific tool to verify the soundness and completeness of the
algorithm description. Compilation, execution and redistribution
of this file may violate patents rights in certain countries.
The situation being different for every country and changing
over time, it is your responsibility to determine which patent
rights restrictions apply to you before you compile, use,
modify, or redistribute this file. A patent lawyer is qualified
to make this determination.
```

```
If and only if they don't conflict with any patent terms, you
can benefit from the following license terms attached to this
file.
```

3.3.3 3.3. License

Every source code file **must** mention a usage and redistribution license after the copyright attribution (and patent warning for algorithms potentially linked to a patent). Of course, if the authors use or modify a file previously written by other persons, the license chosen by the previous authors **must not** be modified.

- When the authors are not aware of a possible patent issue, the license **must** be a free software license of the GPL[34]/LGPL[35]/AGPL[36] or BSD[37] type.
- When a source code file can be linked to a patented algorithm and the source code authors are not the patent inventors, the file **must** be distributed under the BSD license.
- When a source code file can be linked to a patented algorithm and the authors of the source code are the patent inventors, the file **must** be distributed either under the BSD license or "for research and education only"[38].

These wordings are **recommended**:

```
This program is free software: you can use, modify and/or
redistribute it under the terms of the GNU General Public
License as published by the Free Software Foundation, either
version 3 of the License, or (at your option) any later
version. You should have received a copy of this license along
this program. If not, see <http://www.gnu.org/licenses/>.

This program is free software: you can use, modify and/or
redistribute it under the terms of the simplified BSD
License. You should have received a copy of this license along
this program. If not, see
<http://www.opensource.org/licenses/bsd-license.html>.
```

[34] `http://www.gnu.org/licenses/gpl.html`
[35] `http://www.gnu.org/licenses/lgpl.html`
[36] `http://www.gnu.org/licenses/agpl.html`
[37] `http://www.opensource.org/licenses/bsd-license.php`
[38] Distribution "for research and education" can help avoid conflicts between patent rights and software license when the patent inventors are the source code authors. It is not needed in other situations because the validity of the license will depend on the local patent regulations, as stated in the last sentence of the patent warning.

```
This program is provided for research and education only: you can
use and/or modify it for these purposes, but you are not allowed
to redistribute this work or derivative works in source or
executable form. A license must be obtained from the patent right
holders for any other use.
```

The exact terms can differ, for example when a different GPL/LGPL/AGPL license version is chosen. The full text of the license **must** be included in a separate file with the source code.

IPOL authors **should** verify that the usage of these licenses for their software publications complies with their employer policy and local situation and jurisdiction.

3.4 4. Documentation

3.4.1 4.1. README.txt

Every IPOL program **must** provide a file named `README.txt` in the base folder and written in plain text and in English. This `README.txt` file **must** include the following essential information, in any order:

- name and brief description of the program
- reference to the IPOL article
- authors and contact information
- version number and release date
- location of future releases and updates
- copyright, patent and license information
- tools and libraries needed to compile and use the program
- compilation instructions
- usage instructions and example
- changes in the program since it was first published in IPOL

This `README.txt` file **may** contain other information, and **may** also be completed by another documentation, possibly with more details, in text, PDF, HTML or any other format.

For a simple code, the license information in `README.txt` can be

```
This program is written by Jane Doe <jane.doe@example.org> and
distributed under the terms of the GPLv3 license.
```

A complex case (multiple authors, patents and licenses) can be:

```
This program is written by Taro Yamada <taro.yamada@example.jp>
and Juan Perez <juan.perez@example.es> with contributions from
Marie Untel, ENS Cachan <marie.untel@ens-cachan.fr>.
- mmatch.c and rot_tree.c may be linked to the pending EU patent
  123.456 by Taro Yamada and Juan Perez and are provided for
  scientific and education only.
- demoz.c may be linked to the US patent 65.43.21 by Jane Doe;
  see the file for license terms.
- eizo.c and linalg_lib.c are distributed under the terms
  of the BSD license.
- All the other files are distributed under the terms of the
  LGPLv3 license.
```

3.4.2 4.2. Readability

In an IPOL program, the source code is a primary material for the publication. It will be reviewed, published and read like any other part of the article. The authors **must** take care of the clarity of their program.

The source code of an IPOL program **must** be consistently indented and spaced. Lines **should** be limited to 80 characters and **should not** end with blank characters (spaces, tabs, ...). Files **should not** have more than 1000 lines. The line terminations **should** be the same (DOS/Windows `CRLF` or UNIX `CR` style) for all the files of the program.

Functions **should** be grouped by abstraction level in different source code files:

- the `main()` function, command-line processing and input/output calls in one file,
- the implementation of the algorithm described and reviewed in the IPOL article in one or more other files,
- and the implementation of auxiliary and external routines in one or more other files.

Annex D provides some examples of programs that can be used to improve the indentation, spacing and presentation of a source code.

3.4.3 4.3. Implementation and Comments

The source code of an IPOL program **must** be commented precisely and exhaustively. Authors **should** target the "1/8 comment/instruction ratio", but the quality of the comments is more important than the quantity. The source code **must** be written in English, including all variables, functions names and comments.

Authors **must** ensure that the code is understandable, to the satisfaction of the editor and reviewers, so that consistency between the description of the algorithm and its implementation can be verified. The relation between each part of the implementation and the respective part of the description of the algorithm **must** be explained in the comments.

Authors **should** apply simpler implementations when available, follow the conventions of the programming language, and use comments to explain implementation choices and every complicated or subtle point in the program. Clarity is more important than virtuosity.

Every function **must** be documented with at least one line explaining what the function is doing, and the meaning of its parameters and return value.

The Doxygen[39] source code documentation format is **recommended** for every IPOL Program.

Annex D provides some examples of programs that can be used to count the comment, instruction and blank lines.

3.4.4 4.4. Example Data

The authors **should** provide an example of input file to test the IPOL program and the result to expect when this input file is processed by the program.

4 Annexes

The annexes are not part of the guidelines. They are provided to help authors and editors follow the guidelines.

4.1 A. Key Words

The key words **must**, **must not**, **required**, **shall**, **shall not**, **should**, **should not**, **recommended**, **may**, and **optional** in this document are to be interpreted as described in IETF RFC2119[40].

[39] `http://www.stack.nl/~dimitri/doxygen/`

[40] `http://tools.ietf.org/html/rfc2119`

must This word, or the terms **required** or **shall**, mean that the definition is an absolute requirement.

must not This phrase, or the phrase **shall not**, mean that the definition is an absolute prohibition.

No article will be published in IPOL if a program included in this article doesn't follow such requirements or prohibitions.

should This word, or the adjective **recommended**, mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.

should not This phrase, or the phrase **not recommended** mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.

The rationale for not following such recommendations **must** be agreed by the authors, reviewers and editor before an article is accepted for publication.

may This word, or the adjective **optional**, mean that an item is truly optional. One author may choose to include the item because a particular article requires it or because the author feels that it enhances the software while another author may omit the same item.

4.2 B. Compression and Archive Tools

IPOL authors can consider that the files produced by the following tools are correct zip and tar/gzip archives:

- on Linux systems, the `tar` (usually "GNU tar"), `zip` (usually "Info-Zip zip") and `gzip` programs;
- on Mac OS X systems, the "Create Archive" feature of the graphical interface and the `tar` (usually "BSD tar") and `gzip` programs;
- on Windows systems, the "Compressed Folder" feature of the graphical interface and the `7-zip`[41] program.

[41] `http://www.7-zip.org/`

4.3 C. Coding Help

The authors of the IPOL article do not need to be the authors of all the implementation of their algorithm. They can use their own code or code from other IPOL programs or other software projects and libraries, or a combination of these options. This is encouraged when it helps improve the quality of the implementation. All the source code must follow the guidelines (be standard, portable, readable and documented), regardless of its origin. The original copyrights and licenses of the reused code parts must of course be respected.

Some help to read and write image files is provided in the IPOL wiki[42] with simplified interfaces to `libpng` and `libtiff` and some examples. IPOL authors can also find in this wiki a list of contributions and tools[43] from IPOL authors willing to share.

The IPOL editorial board can provide some help to the authors to accelerate their code and guide them for performance profiling and parallel programming. The IPOL discussion list[44] is the good place for any questions related to IPOL, including the implementation work.

4.4 D. Source Code Tools

IPOL authors can use these programs to improve the presentation and clarity of their source code: `indent`[45] (C, Linux), `uncrustify`[46] (C/C++, Linux and Windows), `astyle`[47] (C/C++, Linux, Mac OS X and Windows) and `UniversalIndentGUI`[48] (cross-platform graphical frontend).

Tools like `cloc`[49], `ohcount`[50] and `sloccount`[51] can be used to count the comments, instructions and blank lines and evaluate the comment/instruction ratio.

[42] `http://tools.ipol.im/wiki/author/code/tools/`
[43] `http://tools.ipol.im/wiki/author/code/hatchery/`
[44] `http://tools.ipol.im/mailman/listinfo/discuss`
[45] `http://www.gnu.org/software/indent/`
[46] `http://uncrustify.sourceforge.net/`
[47] `http://astyle.sourceforge.net/`
[48] `http://universalindent.sourceforge.net/`
[49] `http://cloc.sourceforge.net/`
[50] `http://ohcount.sourceforge.net/`
[51] `http://sloccount.sourceforge.net/`