Ion: Notes for the module and patch writer

Tuomo Valkonen
tuomov at iki.fi

February 23, 2007

Abstract

This document is an unorganized collection of notes for those who want to write modules or patches to Ion.

Contents

1 Class and object hierarchies

While Ion does not not have a truly object-oriented design [1], things that appear on the computer screen are, however, quite naturally expressed as such "objects". Therefore Ion implements a rather primitive OO system for these screen objects and some other things.

It is essential for the module writer to learn this object system, but also people who write their own binding configuration files necessarily come into contact with the class and object hierarchies – you need to know which binding setup routines apply where, and what functions can be used as handlers in which bindings. It is the purpose of this section to attempt to explain these hierarchies. If you do not wish the read the full section, at least read the summary at the end of it, so that you understand the very basic relations.

For simplicity we consider only the essential-for-basic-configuration Ioncore, *mod_tiling* and *mod_query* classes. See Appendix **??** for the full class hierachy visible to Lua side.

1.1 Class hierarchy

One of the most important principles of object-oriented design methodology is inheritance; roughly how classes (objects are instances of classes) extend on others' features. Inheritance gives rise to class hierarchy. In the case of single-inheritance this hierarchy can be expressed as a tree where the class at the root is inherited by all others below it and so on. Figure 1 lists out the Ion class hierarchy and below we explain what features of Ion the classes implement.

The core classes:

Obj Is the base of Ion's object system.

WRegion is the base class for everything corresponding to something on the screen. Each object of type WRegion has a size and position relative to the parent WRegion. While a big part of Ion operates on these instead of more specialised classes,

1. the author doesn't like such artificial designs

```
Obj
 |-->WRegion
 |    |-->WClientWin
 |    |-->WWindow
 |    |    |-->WMPlex
 |    |    |    |-->WFrame
 |    |    |    |-->WScreen
 |    |    |         |-->WRootWin
 |    |    |-->WInput (mod_query)
 |    |         |-->WEdln (mod_query)
 |    |         |-->WMessage (mod_query)
 |    |-->WGroup
 |    |    |-->WGroupWS
 |    |    |-->WGroupCW
 |    |-->WTiling (mod_tiling)
 |-->WSplit (mod_tiling)
```

Figure 1: Partial Ioncore, *mod_tiling* and *mod_query* class hierarchy.

WRegion is a ”virtual” base class in that there are no objects of ”pure” type WRegion; all concrete regions are objects of some class that inherits WRegion.

WClientWin is a class for client window objects, the objects that window managers are supposed to manage.

WWindow is the base class for all internal objects having an X window associated to them (WClientWins also have X windows associated to them).

WRootWin is the class for root windows of X screens. Note that an ”X screen” or root window is not necessarily a single physical screen as a root window may be split over multiple screens when hacks such as Xinerama are used. (Actually there can be only one root window when Xinerama is used.)

WMPlex is a base class for all regions that”multiplex” other regions. This means that of the regions managed by the multiplexer, only one can be displayed at a time. Classes that inhereit WMPlex include screens and frames.

WScreen is the class for objects corresponding to physical screens. Screens may share a root window when the Xinerama extension is used as explained above.

WFrame is the class for frames. While most Ion's objects have no graphical presentation, frames basically add to WMPlexes the decorations around client windows (borders, tabs).

WGroup is the base class for groups. Particular types of groups are workspaces (WGroupWS) and groups of client windows (WGroupCW).

Classes implemented by the *mod_tiling* module:

WTiling is the class for tilings of frames.

WSplit (or, more specifically, classes that inherit it) encode the WTiling tree structure.

Classes implemented by the *mod_query* module:

```
WRootWins
 |-->WScreens
      |-->WGroupWSs
      |-->WTilings
      |-->WClientWins in full screen mode
      |-->WFrames
           |-->WGroupCWs
           |-->WClientWins
           |-->WFrames for transients
           |-->a possible WEdln or WMessage
```

Figure 2: Most common parent–child relations

WInput is a virtual base class for the two classes below.

WEdln is the class for the ”queries”, the text inputs that usually appear at bottoms of frames and sometimes screens. Queries are the functional equivalent of ”mini buffers” in many text editors.

WMessage implements the boxes for warning and other messages that Ion may wish to display to the user. These also usually appear at bottoms of frames.

There are also some other ”proxy” classes that do not refer to objects on the screen. The only important one of these for basic configuration is WMoveresMode that is used for binding callbacks in the move and resize mode.

1.2 Object hierarchies: WRegion parents and managers

1.2.1 Parent–child relations

Each object of type WRegion has a parent and possibly a manager associated to it. The parent for an object is always a WWindow and for WRegion with an X window (WClientWin, WWindow) the parent WWindow is given by the same relation of the X windows. For other WRegions the relation is not as clear. There is generally very few restrictions other than the above on the parent—child relation but the most common is as described in Figure 2.

WRegions have very little control over their children as a parent. The manager WRegion has much more control over its managed WRegions. Managers, for example, handle resize requests, focusing and displaying of the managed regions. Indeed the manager—managed relationship gives a better picture of the logical ordering of objects on the screen. Again, there are generally few limits, but the most common hierarchy is given in Figure 3. Note that sometimes the parent and manager are the same object and not all objects may have a manager (e.g. the dock in the dock module at the time of writing this) but all have a parent–a screen if not anything else.

```
WRootWins
 |-->WScreens
      |-->WGroupCWs for full screen WClientWins
      |    |-->WClientWins
      |    |-->WFrames for transients (dialogs)
      |         |--> WClientWin
      |-->WGroupWSs for workspaces
      |    |-->WTiling
      |    |    |-->possibly a WEdln, WMessage or WMenu
      |    |    |-->WFrames
      |    |         |-->WGroupCWs (with contents as above)
      |    |-->WFrames for floating content
      |-->WFrames for sticky stuff, such as the scratchpad
```

Figure 3: Most common manager–managed relations

1.2.2 Manager–managed relations

Note that a workspace can manage another workspace. This can be achieved with the `attach_new` function, and allows you to nest workspaces as deep as you want.

1.3 Summary

In the standard setup, keeping queries, messages and menus out of consideration:

- The top-level objects that matter are screens and they correspond to physical screens. The class for screens is WScreen.
- Screens contain (multiplex) groups (WGroup) and other objects, such as WFrames. Some of these are mutually exclusive to be viewed at a time.
- Groups of the specific kind WGroupWS often contain a WTiling tiling for tiling frames (WFrame), but groups may also directly contain floating frames.
- Frames are the objects with decorations such as tabs and borders. Frames contain (multiplex) among others (groups of) client windows, to each of which corresponds a tab in the frame's decoration. Only one client window (or other object) can be shown at a time in each frame. The class for client windows is WClientWin.

2 Object system implementation

First, to get things clear, what are considered objects here are C structures containing a properly initialized structure defined in *ioncore/obj.h* as the first element (or the first element of the structure which is the first element and so on which gives rise to inheritance). The WObj structure contains a pointer to a WObjDescr class type info structure and a list of so called "watches". The WObjDescr structure simply lists the class name, a table of dynamic functions and a pointer to deinitialisation function (or ”destructor”).

Ion does not do any reference counting, garbage collecting or other fancy things related to automatic safe freeing of objects with its simplistic object system. Instead special watches (the WWatch structure) may be used to create safe references to objects that might be destroyed during the time the specific pointer is needed. When an object is destroyed, its list of watches is processed, setting the pointers in the watches to NULL and the watch handlers for each watch are called.

3 The Lua interface

This section finally describes the implementation details and how modules should us the Lua interface. First, in section 3.1 we look at types supported by the interface, how objects are passed to Lua code and how Lua tables should be accessed from Ion and modules. In section 3.2 the methods for exporting functions and how they are called from Lua are explained and in section 3.3 the method for calling Lua functions is explained.

3.1 Supported types

The following types are supported in passing parameters between the C side of Ion and Lua:

Identifier character	C type	Description
i	int	Integer
s	char*	String
S	const char*	Constant string
d	double	
b	bool	
t	ExtlTab	Reference to Lua table
f	ExltFn	Reference to Lua function.
o	Any WObj*	

The difference between identifiers ’s’ and ’S’ is that constant strings as return values are not free’d by the level 1 call handler (see below) after passing to Lua (`lua_pushstring` always makes a copy) unlike normal strings. String parameters are always assumed to be the property of the caller and thus implicitly const.

Likewise, if a reference to ’t’ or ’f’ is wished to be stored beyond the lifetime of a function receiving such as an argument, a new reference should be created with `extl_ref_table` `/fn`. References can be free’d with `extl_unref_table/fn`. References gotten as return values with the `extl_table_get` (how these work should be self-explanatory!) functions are property of the caller and should be unreferenced with the above-mentioned functions when no longer needed. The functions `extl_fn/table_none()` return the equivalent of NULL.

WObjs are passed to Lua code with WWatch userdatas pointing to them so the objects can be safely deleted although Lua code might still be referencing them. (This is why SWIG or tolua would not have helped in creating the interface: extra wrappers for each function would still have been needed to nicely integrate into Ion's object system. Even in the case that Ion was written in C++ this would be so unless extra bloat adding pointer-like objects were used everywhere instead of pointers.) It may be sometimes necessary check in Lua code that a value known to be an Ion WObj is of certain type. This can be accomplished with `obj_is(obj, "typename")`. `obj_typename(obj)` returns type name for a WObj.

3.2 Exporting functions

Exported functions (those available to the extension language) are defined by placing `EXTL_EXPORT` before the function implementation in the C source. The script mkexports.pl is then used to automatically generate *exports.c* from the source files if `MAKE_EXPORTS=modulename` is specified in the Makefile. All pointers with type beginning with a 'W' are assumed to be pointers to something inheriting WObj. In addition to a table of exported functions and second level call handlers for these, *exports.c* will contain two functions `module_register_exports()` and `module_unregister_exports()` that should then be called in module initialisation and deinitialisation code.

You've seen the terms level 1 and 2 call handler mentioned above. The Lua support code uses two so called call handlers to convert and check the types of parameters passed from Lua to C and back to Lua. The first one of these call handlers is the same for all exported functions and indeed lua sees all exported as the same C function (the L1 call handler) but with different upvalues passing a structure describing the actual function and the second level call handler. The L1 call handler checks that the parameters received from Lua match a template given as a string of the identifier characters defined above. If everything checks out ok, the parameters are then put in an array of C unions that can contain anyof these known types and the L2 call handler is called.

The L2 call handler (which is automatically generated by the mkexports.pl script) for each exported function checks that the passed WObjs are of the more refined type required by the function and then calls the actual function. While the WObj checking could be done in the L1 handler too, the L2 call handlers are needed because we may not know how the target platform passes each parameter type to the called function. Thefore we must let the C compiler generate the code to convert from a simple and known enough parameter passing method (the unions) to the actual parameter passing method. When the called function returns everything is done in reverse order for return values (only one return value is supported by the generated L2 call handlers).

3.3 Calling Lua functions and code

The functions `extl_call`, `extl_call_named`, `extl_dofile` and `extl_dostring` call a referenced function (ExtlFn), named function, execute a string and a file, respectively.

The rest of the parameters for all these functions are similar. The 'spec' argument is a string of identifier characters (see above) describing the parameters to be passed. These parameters follow after 'rspec'. For dofile and dostring these parameters are passed in the global table arg (same as used for program command lien parameters) and for functions as you might expect. The parameter 'rspec' is a similar description of return values. Pointers to variables that should be set to the return values follow after the input values. The return value of all these functions tells if the call and parameter passing succeeded or not.

Sometimes it is necessary to block calls to all but a limited set of Ion functions. This can be accomplished with `extl_set_safelist`. The parameter to this function is a NULL-terminated array of strings and the return value is a similar old safelist. The call `extl_set_safelist(NULL)` removes any safelist and allows calls to all exported functions.

3.4 Miscellaneous notes

Configuration files should be read as before with the function `read_config_for` except that the list of known options is no longer present.

Winprops are now stored in Lua tables and can contain arbitrary properties. The 'proptab' entry in each WClientWin is a reference to a winprop table or `extl_table_none()` if such does not exist and properties may be read with the `extl_table_gets` functions. (It is perfectly legal to pass `extl_table_none()` references to `extl_table_get*`.)

4 Miscellaneous design notes

4.1 Destroying WObj:s

To keep Ion's code as simple as possible yet safe, there are restrictions when the WObj `destroy_obj` function that calls watches, the deinit routine and frees memory may be called directly. In all other cases the `defer_destroy` function should be used to defer the call of `destroy_obj` until Ioncore returns to its main event loop.

Calling the `destroy_obj` function directly is allowed in the following cases:

- In the deinit handler for another object. Usually managed objects are destroyed this way.
- The object was created during the current call to the function that wants to get rid of the object. This is the case, for example, when the function created a frame to manage some other object but for some reason failed to reparent the object to this frame.
- In a deferred action handler set with `defer_action`. Like deferred destroys, other deferred actions are called when Ioncore has returned to the main loop.
- You are absolute sure that C code outside your code has no references to the object.

If there are no serious side effects from deferring destroying the object or you're unsure whether it is safe to destroy the object immediately, use `defer_destroy`.

4.2 The types `char*` and `const char*` as function parameters and return values

The following rules should apply to using strings as return values and parameters to functions.

Type	Return value	Parameter
`const char*`	The string is owned by the called function and the caller is only quaranteed short-term read access to the string.	The called function may only read the string during its execution. For further reference a copy must be made.
`char*`	The string is the caller's responsibility and it *must* free it when no longer needed.	The called function may modify the string but the "owner" of the string is case-dependant.

5 C coding style

If you want to submit patches to Ion, you MUST follow my coding style, even if you think it is the root of all evil. We don't want the code to be an incomprehensible mess of styles and I have better things to do than fix other people's style to match mine. The style should be obvious by studying the source, but here's a list of some things to take note of.

5.1 Whitespace

- Indentations of 4 with *tab size=4*.
- No extra spaces between operators, delimiters etc. except
  - around logical and, or (`&&`, `||`)
  - around the conditional `a ? b : c`
  - after commas and semicolons

  In my opinion this helps pointing out arithmetic or other expressions within logical expressions or parameter lists.
- All kinds of labels are out-tended to the level of the higher level block. For example:

```
void foo()
{
again:
    switch(asdf){
    case 1:
        ...
        break;
```

```
    default:
        ...
        break;
    }
}
```

5.2 Braces

- Opening brace is at the end of the line, except in function bodies, where it is at the beginning of the line following the definition.
- Never put the body of a control statement on the same line with the statement (e.g. `if(foo){ bar() }`).
  For example, the block
  ```
  void foo(int a, int b)
  {
      if(a==b && c+d==e){
          ...
      }
  }
  ```
  has correct style while the block
  ```
  void foo(int a,int b) {
      if (a == b && c + d == e) {
          ...
      }
  }
  ```
  does not.
- The `else` keyword follows immediately after the closing brace of previous `if`, if any. (This might change so I don't care if you put it on the next line.)
- I have used the convention that control statement bodies containing a single statement do not need braces around the block if, in case of the `if` all the blocks in `if ... else if ... else` contain just one statement. If you want to, just use braces in every case.

5.3 Names

- Function and variable names only have lower case letters. Type names are in mixed case while constants and macros (`#define`s) are in upper case letters.

5.4 Miscellaneous

- In the definition of a pointer variable, the asterisk is attached to the variable name: `char *s;`. (One could claim this an exception to the second rule.)
- You might optionally want to use Jed's foldings to group blocks of related code in a file to keep it organized:

```
/*{{{ Many related functions */

void code()
{
    ...
}

...

/*}}}*/
```

I think that's mostly it. Study the source when in doubt.

A The GNU General Public License

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software—to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

Terms and Conditions For Copying, Distribution and Modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".
   Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.
1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.
2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

(a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
(b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
(c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:
   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or exe-

cutable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.
5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.
6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.
7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

   If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

No Warranty

11. Because the program is licensed free of charge, there is no warranty for the program, to the extent permitted by applicable law. Except when otherwise stated in writing the copyright holders and/or other parties provide the program "as is" without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the program is with you. Should the program prove defective, you assume

THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Appendix: How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

> one line to give the program’s name and a brief idea of what it does.
> Copyright (C) yyyy name of author
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
> This program 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. See the GNU General Public License for more details.
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111-1307, USA.

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) yyyy name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.

> This is free software, and you are welcome to redistribute it under certain conditions; type 'show c' for details.

The hypothetical commands `show w` and `show c` should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w` and `show c`; they could even be mouse-clicks or menu items—whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program 'Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> signature of Ty Coon, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Index