Lisp in Summer Projects Submission

Submission Date	2013-10-03 12:39:10
Full Name	Ryan Pavlik
Country	United States
Project Name	cl-autowrap
Type of software	library
General category	other
LISP dialect	Commmon Lisp
GitHub URL	https://github.com/rpav/cl-autowrap
Did you start this project?	Yes, all the code is written by me
Project Description	I want to describe my project in this form.
Purpose	cl-autowrap automates the generation of complete FFI wrappers for C libraries using c2ffi and given a ".h" header file. It also implements a new, simplified higher-level FFI on top of CFFI-SYS. Wrapper authors can thus concentrate on the "lispy" side of their API rather than maintaining low-level definitions by hand.
Function	Autowrap calls c2ffi (https://github.com/rpav/c2ffi) which parses and generates JSON spec files for input C. It then takes this information and generates a complete-as-possible set of accessors and "safe"-but-thin wrappers for records, types, functions, constants, externs, etc.
Motivation	SWIG is broken beyond use. Wrapping things by hand is more work and more error-prone than it should be. CFFI's higher-level interface also is not terribly suited to generated wrappers. I had written c2ffi and a prior wrapper generator, c2ffi-cffi, but this didn't prove terribly useful in practice while trying to wrap SDL2. Thus, cl-autowrap was born, and https://github.com/lispgames/cl-sdl2 is one of a growing number of projects which put it to practical use.
Audience	Anyone who wants to provide FFI wrappers for C libraries in

	Common Lisp.
Methodology	Autowrap implements SFFI, a "simplified FFI" which provides a fairly complete, if pragmatic, model of C's declarations. While these are not particularly complicated, providing an accurate model for unusual-but-legal cases is necessary, as these tend to crop up regularly. Examples include anonymous record members, type aliases with names identical to the types aliased, or symbols which differ only by case. This is built on a few basic functions provided by the lower-level CFFI-SYS package: allocate, free, memory-set, memory-get, and foreign-funcall (not exact names). CFFI-SYS was chosen since it already handled a number of Common Lisp implementations, and provided the core set of functions necessary for a higher-level interface. Autowrap calls and parses the (JSON) output from c2ffi, and writes appropriate SFFI declarations. Also provided are a number of convenience functions for handling common things such as enums, bitmasks, and the like.
Conclusion	This is already fairly complete. CL-SDL2 provides an in-practice example of its usage. Other projects are currently being wrapped and Autowrap is constantly being improved, both to make it easier to use and to fix any bugs which are encountered. Currently, I'm working on adding C++ output to c2ffi, followed by the "cxx2c" project, which will autogenerate a C library with class and function wrappers with well-defined mangled names. At this time, cl-autowrap will be extended to support better syntax for calling C++ using these intermediate libraries. Note: while I started and have written the majority of this project (cl-autowrap), it has received contributed code, most notably from Samium Gromoff (https://github.com/deepfire). This has been a project primarily of necessity and pragmatism; I did not consider submitting it until literally the last day of registration.
Build Instructions	Get the latest desired revision from github; link the ASD somewhere your ASDF can find it. Then: (asdf:load-system 'cl-autowrap) This does not technically require c2ffi: once a project has ".spec" files generated, no further use of c2ffi is necessary. However, if you wish to wrap something, you will need llvm/clang of at least version 3.3, and c2ffi: https://github.com/rpav/c2ffi
Test Instructions	You may attempt to wrap a C library following the instructions in the README, if you have installed c2ffi. Additionally or alternatively, you may load a project such as CL-SDL2 which uses autowrap for definitions.

Execution Instructions	This is a library project, and execution is not relevant.
Describe any bugs or caveats	As of the Sept 30 cutoff, some known bugs may apply; it's possible that C constants such as LDBL_MAX will not parse due to an oversight in the fix. Additionally, c2ffi may produce a bare "inf" instead of a quoted "inf", thus producing illegal JSON for such values. This has since been fixed. Additionally, c2ffi has only really been tested on x86_64 Linux. However, output has been confirmed as working on 32-bit architectures for at least CL-SDL2.
Official	I have read rules and have abided by them. I am 18 years of age or older. I am not living in Brazil, Quebec, Saudi Arabia, Cuba, Iran, Myanmar (Burma), North Korea, Sudan, or Syria.