**The Problem**
- There are over 20 different coding styles for C.
- Various open source projects use different styles.
- Most programmers have their own favourite style.
- This raises the barrier for entry to an open source project and negates some of the benefits of having a defined style in the first place.
- Most of them differ over similar aspects: naming scheme, white space, etc.

**The Solution**
- **CELS**, a tool which allows interconversion between different coding styles.
- CELS allows the user to convert to/from a set of predefined styles, such as FreeBSD, Linux and GNU.
- The user can create a specification for their own, preferred style.
- These features together allow users to contribute to a project in a foreign style, without getting bogged down adjusting to the new style.
- CELS is based on the stable libc++ API, so doesn't depend on a specific version of LLVM.

**Pattern Based Symbol Renaming**
- CELS can perform a pattern based rename, bringing an entire code collection into line with a specified coding standard.
- Different Naming Schemes can be specified per symbol type.
- Works across multiple files.
- Only renames symbols whose definition is given in the sources - so it ignores symbols defined in external libraries and APIs.
- Allows user defined patterns/naming schemes.
- Keeps track of inconsistent symbol names in the original document - so a conversion to a naming scheme and back is lossless.

**White Space**
- CELS can deal with the white space aspects of code formatting, including:
  - Dealing with indents (better than existing tools - correctly handles macros)
  - Correcting inter-symbol spacings, such as "int " and "int x;
  - Creating pretty alignment (eg, aligning function names with different return types).

**Optional Line Breaking**
- CELS uses a custom-made dynamic-programming algorithm (inspired by TeX) to insert line breaks in the optimal places.
- Each pair of tokens is assigned a line breaking penalty.
- The total penalty over the set of lines is minimized.
- The user can define the scores for pairs of tokens.

**Consistency Checking**
- CELS is capable of checking a code base to ensure that it meets a certain style, giving a list of errors for every point the style is violated.
- This is designed to be integrated with open source projects, so the more mundane aspects of code review can be automated.

**Style Inference**
- CELS can be run over a code base to infer the coding style used.
- Then, by using the Checker, CELS can report occasions when the style is not followed, or by using the Converter, CELS can correct the violations.

**Future Plans**
- **Additional Features:**
  - Conversion between typedefs and Hungarian (eg, "int _verbose;") to "typedef flag int;
  - flag verbose;")
  - Extension to support Objective-C and C++.
- **Integration:**
  - With a VCS, so local code can be converted to project style when committed/pushed.
- The source code is available under the FreeBSD license at: https://bitbucket.org/PEConn/cecls