

Coming Soon to Allegro CL, Streams: Faster, Simpler, and International

Charles A. Cox
Franz Inc.

References: [1] <http://www.franz.com/support/docs/simple-stream.htm>
[2] [Future] Allegro CL 6.0 Documentation

18 May 2000 Franz Inc. 1

Outline

- *Introduction*
- International Character Support
 - Unicode
 - External-Formats
 - Unicode ↔ External-Formats
 - Dynamism
 - Composability
- Bivalent Streams
 - Background
 - ANSI Common Lisp Support Issues
 - Allegro CL Simple Streams

- Allegro CL Simple Streams Description
 - 'Gray'-Streams Replacement
 - User Level / Strategy Level / Device Level
- Conclusion
 - Performance Results

18 May 2000 Franz Inc. 2

Introduction

- Highlight upcoming Allegro CL 6.0 changes.
- Changes are partly user-visible.
- Changes diverge (slightly) from ANSI Specification.

18 May 2000 Franz Inc. 3

Introduction (cont.)

- Goals:
 - Integrate International Character support into streams; in particular, to allow dynamic external-format switching.
 - To simplify low-level streams implementation, thereby speeding up basic functionality.
 - To meet the needs of modern Internet and Interoperability applications.

18 May 2000 Franz Inc. 4

International Character Support

- Background:
 - Asian language characters require more than the 8-bits traditionally allocated for (C) programming language representation.
 - Various "multi-byte" representations exist.
 - ASCII characters occupy 8-bits each.
 - Adjacent non-ASCII characters occupy several 8-bit bytes.

18 May 2000 Franz Inc. 5

International Character Support (cont.)

Example: 'Tokyo is 東京'

ASCII characters occupy 1 octet each.

Japanese characters occupy 2 octets each.

Shift-JIS: 54 6f 79 6f 20 69 73 20 93 8c | 8b 9e

T o k y o <sp> i s <sp> 東 京

18 May 2000 Franz Inc. 6

International Character Support (cont.)

- Problems with using “multi-byte” representations internally.
 - Several Different encodings exist for same character set (eg, Japanese has EUC, JIS, Shift-JIS, etc.)
 - Random character indexing is difficult.

18 May 2000

Franz Inc.

7

International Character Support (cont.)

- Example:
 - “Tokyo is 東京”

| | | | | | | | | | | | | | |
|------------|----|----|----|----|----|------|----|----|------|----|----|----|----|
| | T | o | k | y | o | <sp> | i | s | <sp> | 東 | 京 | | |
| | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | | |
| Shift-JIS: | 54 | 6f | 6b | 79 | 6f | 20 | 69 | 73 | 20 | 93 | 8c | 8b | 9e |
| EUC: | 54 | 6f | 6b | 79 | 6f | 20 | 69 | 73 | 20 | c5 | ec | b5 | f3 |
| JIS: | 54 | 6f | 6b | 79 | 6f | 20 | 69 | 73 | 20 | 1b | 24 | 42 | 45 |
| | | | | | | | | | | 6c | 35 | 7e | |

18 May 2000

Franz Inc.

8

International Character Support (cont.)

- International Allegro CL (IACL), first release 1990, represents Lisp characters internally using exactly 16-bits each.
- Unix IACL 5.0.1 uses Process-Code, Japanese-centric.
- Windows IACL 5.0.1 uses Unicode.
- IACL 6.0, *all platforms*, use Unicode.

18 May 2000

Franz Inc.

9

Unicode Representation

- Standard, fixed-width, uniform encoding for written characters and text.
- Includes technical symbols and characters for the major scripts of the world.
- Unicode encoding treats alphabetic characters, ideographic characters, and symbols identically. — No escape sequence or control code used to specify any character in any language.

10

Unicode Representation (cont.)

- Example:
 - “Tokyo is 東京”

| | | | | | | | | | | | |
|----------|------|------|------|------|------|------|------|------|------|------|------|
| Unicode: | 0054 | 006f | 006b | 0079 | 006f | 0020 | 0069 | 0073 | 0020 | 6771 | 4eac |
| | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ | ↓ |
| | T | o | k | y | o | <sp> | i | s | <sp> | 東 | 京 |

18 May 2000

Franz Inc.

11

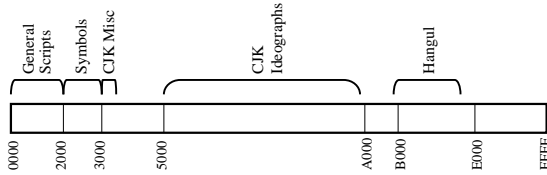
Unicode Representation (cont.)

- Unicode is a 16-bit superset of “Latin-1”, which itself is an 8-bit superset of ASCII.
 - First 256 characters of Unicode are the Latin-1 characters.
- Other 8-bit character encodings have different encodings in Unicode.
 - Example: “Latin Capital Letter L With Stroke”
 - Latin-2 (ISO 8859-2) Value: #xa3
 - Unicode Value: #x0141

12

Unicode Representation (cont.)

- Unicode Allocation (Partial Description):



18 May 2000

Franz Inc.

13

Unicode Representation (cont.)

- Asian Language encodings such as JIS-X0208 for Japanese also have different values in Unicode.
 - Example: “Hiragana Letter A (あ)”
 - JIS-X0208: #x2422
 - Unicode: #x3042

18 May 2000

Franz Inc.

14

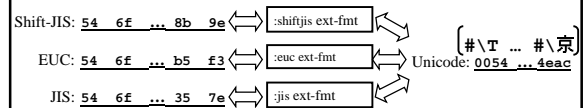
External Formats

- Unicode \Leftrightarrow External Encoding.
 - Allegro CL uses External-Formats to convert character encodings between Unicode and external multi-byte encodings such as ISO 8859-*, EUC, JIS, and Shift-JIS.
 - Several External-Formats are included with Allegro CL 6.0.
 - Users can define new encodings.
 - Definitions are as macros so that conversions take place in-line. (We want I/O to be fast!)

15

External Formats (cont.)

- Examples:
 - “Tokyo is 東京”



18 May 2000

Franz Inc.

16

Unicode \Leftrightarrow External Encoding (cont.)

- Every stream has an associated external-format.
- The external-format is what does the [octets \Leftrightarrow Lisp characters] conversions.
 - The Lisp `read-char` operation may actually input several 8-bit bytes to create a single Unicode character. Similarly, `write-char` may output several 8-bit bytes to represent a multi-byte character.

18 May 2000

Franz Inc.

17

Unicode \Leftrightarrow External Encoding (cont.)

- Users can view/test effect of external-formats using `string-to-octets/octets-to-string` operators.
- Example:

```
> (string-to-octets "東京" :external-format :shiftjis)
#(147 140 139 158 0)
5
> (string-to-octets "東京" :external-format :euc)
#(197 236 181 254 0)
5
```

18 May 2000

Franz Inc.

18

External Formats (cont.)

- Dynamism
 - New in Allegro CL 6.0: A stream's external-format can be changed dynamically.
 - `(setf (stream-external-format stream) external-fmt)`
 - Useful if a socket is being used to transmit/receive characters with multiple encodings.
 - Compiler used at runtime to build high-speed inline external-format converters.
 - Convertors can be pre-compiled.

18 May 2000

Franz Inc.

19

External Formats (cont.)

- Composibility
 - Composed External Format is a special type of external-format which translates between sequences of characters.
 - Can be used to construct/deconstruct Unicode composed characters such as accented characters or ligature characters.

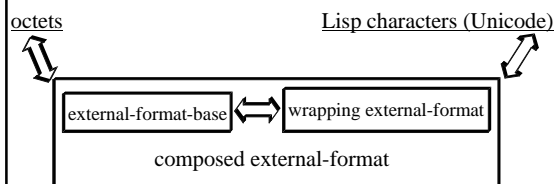
18 May 2000

Franz Inc.

20

External Formats (cont.)

- Example:



18 May 2000

Franz Inc.

21

Composed External Formats

- Example: `#\newline`
 - Windows Text Convention: Each line ends with *two* Ascii characters 'carriage return' (Ascii 13) and 'line feed' (Ascii 10).
 - 'Base Level' external formats convert these to `#\return` and `#\linefeed`.
 - 'Composed Level' external format converts `#\return #\linefeed` combination to/from `#\newline`.

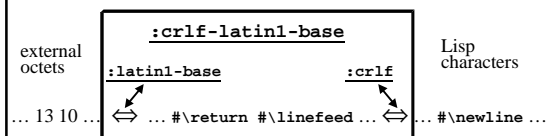
18 May 2000

Franz Inc.

22

Composed External Formats (cont.)

- Example: `#\newline` (cont.)
 - `:crlf-latin1-base` external-format is composed of `:crlf` and `:latin1-base`.



18 May 2000

Franz Inc.

23

Outline

- Introduction
- International Character Support
 - Unicode
 - External-Formats
 - Unicode \leftrightarrow External-Formats
 - Dynamism
 - Composability
- *Bivalent Streams*
 - Background
 - ANSI Common Lisp Support Issues
 - Allegro CL Simple Streams
- Allegro CL Simple Streams Description
 - 'Gray'-Streams Replacement
 - User Level / Strategy Level / Device Level
- Conclusion
 - Performance Results

18 May 2000

Franz Inc.

24

Bivalent Streams

- Background
 - With growth of the Internet and the World Wide Web (specifically, the HTTP protocol), it is desirable to use socket streams for both binary and textual data. Such a stream is called a “bivalent” stream.

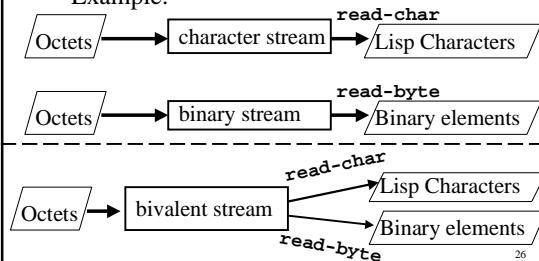
18 May 2000

Franz Inc.

25

Bivalent Streams (cont.)

- Example:



26

Bivalent Streams (cont.)

- ANSI Common Lisp Support Issues
 - ANSI Common Lisp specifically distinguishes *character* and *binary* streams.
 - Character Stream: Created by `open` with `:element-type 'character` [default].
 - Binary Stream: Created by `open` with `:element-type` being a valid array element type, eg, `'float`, or `'(unsigned-byte 8)`, or `'(signed-byte 3)`, etc.

18 May 2000

Franz Inc.

27

Bivalent Streams (cont.)

- ANSI Support Issues (cont.)
 - Note: ANSI Common Lisp defines a *byte* as a contiguous set of bits in an integer. Not necessarily 8-bits wide.
 - `read-byte`/`write-byte` may cause several octets (8-bit bytes) to be read/written.
 - We use *octet* to denote an 8-bit byte.

18 May 2000

Franz Inc.

28

Bivalent Streams (cont.)

- ANSI Support Issues (cont.)
 - File-Position Issues:
 - ANSI Common Lisp specifies that `read-byte`/`write-byte` advance the file-position pointer exactly by “one”. Thus, as defined by ANSI, the file-position pointer *does not necessarily* correspond to the stream octet position.
 - Example:
 - Binary Stream with element-type `(unsigned-byte 16)`: Each `read-byte/write-byte` advances file-position pointer by *two* octets.

18 May 2000

29

Bivalent Streams (cont.)

- Allegro CL Simple Streams
 - The *only* element-type is octet.
 - When `:element-type` is *not* specified in call to `open`, then an Allegro CL Simple Stream is created.
 - When `:element-type` is specified in call to `open`, then a backward-compatible (gray) stream is created.

18 May 2000

Franz Inc.

30

Bivalent Streams (cont.)

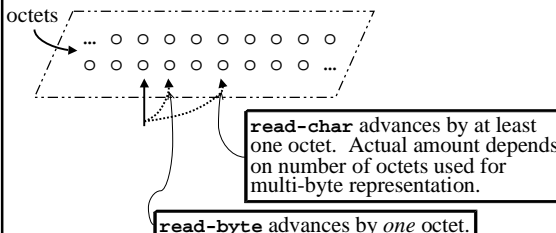
- Every Allegro CL Simple Stream is bivalent.
 - `read-byte/write-byte` operates in terms of octets (one octet per read/write).
 - File-Position is in terms of octets.
- `read-char/write-char` may advance the file-position by more than one in the case of multi-byte external-formats.

18 May 2000

Franz Inc.

31

Bivalent Streams (cont.)

- File-Position Issues Example:
 

32

Outline

- Introduction
- International Character Support
 - Unicode
 - External-Formats
 - Unicode ↔ External-Formats
 - Dynamism
 - Composability
- Bivalent Streams
 - Background
 - ANSI Common Lisp Support Issues
 - Allegro CL Simple Streams
- *Allegro CL Simple Streams Description*
 - ‘Gray’-Streams Replacement
 - User Level / Strategy Level / Device Level
- Conclusion
 - Performance Results

18 May 2000

Franz Inc.

33

Allegro CL Simple Streams Description

- ‘Gray’-Streams Replacement.
 - Background
 - Shortly after the ANSI Common Lisp standardization process, Allegro CL included an implementation of the “Gray Proposal” [for David N. Gray, then of Texas Instruments].

18 May 2000

Franz Inc.

34

‘Gray’-Streams Replacement (cont.)

- Background (cont.)
 - Streams are CLOS objects. Input/Output operations are implemented as methods.
- Problems with Gray Streams
 - Gray Streams distinguish input and output directions per class.
 - Forces combination and mixins in order to model the three different modes (eg, input only, output only, and input/output) for the various stream classes.

35

Problems with Gray Streams (cont.)

- Gray streams methods, which defined the specific streams implementation, are defined immediately below the Common Lisp streams interface level.
 - The CLOS dispatch is performed at higher level than is necessary, thus creating inefficient instruction paths not easily optimizable.
 - The implementation interface of Gray streams overlaps in behavior.

18 May 2000

Franz Inc.

36

Problems with Gray Streams (cont.)

- Example `stream-read-char-no-hang` :
 - The straightforward implementation is to call `stream-read-char` after a call to `stream-listen`.
 - Subclassing a stream, however, can result in a version which does not perform this listen/read combination.

18 May 2000

Franz Inc.

37

Problems with Gray Streams (cont.)

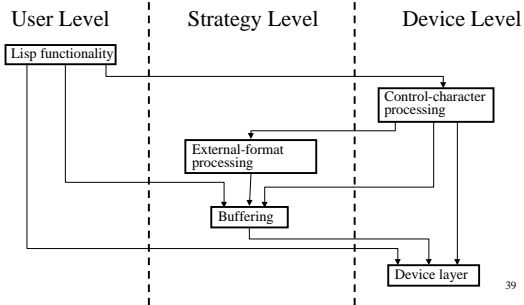
- Example `stream-read-char-no-hang` (cont.):
 - Further subclassing of the stream is not possible without having the source since it is otherwise not possible to know whether to define method for `stream-read-char-no-hang`, `stream-read-char`, `stream-listen`, or perhaps all three.

18 May 2000

Franz Inc.

38

Allegro CL Simple Streams Description (cont.)



39

Allegro CL Simple Streams (cont.)

- User Level
 - `read-char`, `read-sequence`, `write-byte`, etc.
- Device Level
 - Specializable to stream connection type.
 - Level that calls the Operating System and performs buffering.
 - Not intended to be called directly.

18 May 2000

Franz Inc.

40

Conclusions

- Performance Results (Preliminary):
 - write-byte test: 2X speedup
 - Simple Streams Bivalent over Allegro CL 5.0.1 Gray Bivalent (socket streams only): 3X speedup.
 - Character I/O: Time about same, but includes new functionality (character based external-format processing).

18 May 2000

Franz Inc.

41