Making a Language Cross Platform

Libraries and Tooling

Gwen Mittertreiner

Facebook

Goals

- Make you think about cross platform support
- High level, but still have practical advice.

Paths and File Systems

Paths and File Systems

- On Unix, paths are easy to parse right? /usr/bin/easy/peasy

Paths and File Systems

- On Unix, paths are easy to parse right? /usr/bin/easy/peasy
- On Windows, paths are easy

Paths and File Systems

- On Unix, paths are easy to parse right? /usr/bin/easy/peasy
- On Windows, paths are easy to get wrong.

Paths and File Systems

- On Unix, paths are easy to parse right? /usr/bin/easy/peasy
- On Windows, paths are easy to get wrong.
- Which of these are valid paths on Windows?
  - A:\foo\bar
  - B:foo\bar
  - C:/foo/bar
  - D:/foo\bar
  - \\?\E:\foo\bar
  - \\.\F:\foo\bar
  - \??\G:\foo\bar
  - \usr\bin\clang
  - \\?\UNC\abc\xyz
  - \\?\GLOBALROOT\??\UNC\abc\xyz

Paths and File Systems

- On Unix, paths are easy to parse right? /usr/bin/easy/peasy
- On Windows, paths are easy to get wrong.
- Which of these are valid paths on Windows?
  - A:\foo\bar
  - B:foo\bar
  - C:/foo/bar
  - D:/foo\bar
  - \\?\E:\foo\bar
  - \\.\F:\foo\bar
  - \??\G:\foo\bar
  - \usr\bin\clang
  - \\?\UNC\abc\xyz
  - \\?\GLOBALROOT\??\UNC\abc\xyz
- All of them!

Paths and File Systems

- On Unix, paths are easy to parse right? /usr/bin/easy/peasy
- On Windows, paths are easy to get wrong.
- Which of these are valid paths on Windows?
  - A:\foo\bar
  - B:foo\bar
  - C:/foo/bar
  - D:/foo\bar
  - \\?\E:\foo\bar
  - \\.\F:\foo\bar
  - \??\G:\foo\bar
  - \usr\bin\clang
  - \\?\UNC\abc\xyz
  - \\?\GLOBALROOT\??\UNC\abc\xyz
- All of them! * (mostly)

*https://googleprojectzero.blogspot.com/2016/02/the-definitive-guide-on-win32-to-nt.html

Paths and File Systems

Paths and File Systems

- Let system APIs do the lifting

Paths and File Systems

- Let system APIs do the lifting
- Avoid matching paths by streq

Building and Testing

Building and Testing

- Keep cross compiling in mind through out

Building and Testing

- Keep cross compiling in mind through out
- Be very careful with Unix commands

Building and Testing

- Keep cross compiling in mind through out
- Be very careful with Unix commands
- Linker formats

Building and Testing

- Keep cross compiling in mind through out
- Be very careful with Unix commands
- Linker formats
  - Different formats support different things
    - Weak Symbols
    - Symbol Replacement
    - Thread Local Storage
    - DLL Storage
    - Two level namespaces

Porting Existing Libraries

Porting Existing Libraries

- UTF-8 is pretty standard now...

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
  - Define a character type that's different sizes on different platforms

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
  - Define a character type that's different sizes on different platforms
  - Convert everything to a single encoding

Porting Existing Libraries

- UTF-8 is pretty standard now...
    - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
    - Define a character type that's different sizes on different platforms
    - Convert everything to a single encoding
- Be okay with things not making sense

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
  - Define a character type that's different sizes on different platforms
  - Convert everything to a single encoding
- Be okay with things not making sense :(

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
  - Define a character type that's different sizes on different platforms
  - Convert everything to a single encoding
- Be okay with things not making sense :(
- POSIX apis are a handy tool

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
  - Define a character type that's different sizes on different platforms
  - Convert everything to a single encoding
- Be okay with things not making sense :(
- POSIX apis are a handy tool
  - Windows supports many POSIX APIs

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
  - Define a character type that's different sizes on different platforms
  - Convert everything to a single encoding
- Be okay with things not making sense :(
- POSIX apis are a handy tool
  - Windows supports many POSIX APIs
    - With some limitations

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
  - Define a character type that's different sizes on different platforms
  - Convert everything to a single encoding
- Be okay with things not making sense :(
- POSIX apis are a handy tool
  - Windows supports many POSIX APIs
    - With some limitations
  - Android doesn't support some POSIX APIs

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
  - Define a character type that's different sizes on different platforms
  - Convert everything to a single encoding
- Be okay with things not making sense :(
- POSIX apis are a handy tool
  - Windows supports many POSIX APIs
    - With some limitations
  - Android doesn't support some POSIX APIs
    - Depends on API level

Porting Existing Libraries

- UTF-8 is pretty standard now...
  - Except Windows and Apple's Cocoa any sufficiently old platform! Those use UTF-16.
- Two Approaches:
  - Define a character type that's different sizes on different platforms
  - Convert everything to a single encoding
- Be okay with things not making sense :(
- POSIX apis are a handy tool
  - Windows supports many POSIX APIs
    - With some limitations
  - Android doesn't support some POSIX APIs
    - Depends on API level
  - Reimplementing the original target's API set

Thanks!

- I hope you'll all go and make someone porting your project happy.