国際化正規表現ライブラリの開発

日本IBM(株) ソフトウェア開発研究所
長谷川 勇

メニュー

- Background
  - –正規表現とは？
  - –正規表現における国際化
- 既存の実装の問題と本実装による解決
- まとめ
- 今後の課題

(ソフトウェアにおける)正規表現

- 文字列中のパターンを発見/操作する機構
- 様々なテキスト処理ツール(sed, grep, awk, Perl, etc...)で使用
- 例

正規表現	マッチする文字列
abc	: “abc”
a*	: “”, “a”, “aa”, “aaa”, ...
[a-cA]	: “a”, “b” , “c” , “A”

bracket expression

正規表現における国際化

- マルチバイトキャラクタへの対応
  - –従来は日本などに特有の問題
  - –UTF-8の普及によりほとんどの国で必要
- bracket expression
  - –照合要素 1 つとマッチ
  - –複数文字照合要素の問題

照合(Collation)とは?

- 文字列の比較
  - ×コードポイントによる比較(strcmp, wcscmp)
  - ○照合順序による比較 (strcoll, wcscoll)

同じ文字の比較でもロケールに依存して変化

- 文字列を照合要素に分解し、照合要素毎に比較することで行う
  - ただし1文字 == 1照合要素とは限らない

複数文字照合要素

- 複数の文字から成る照合要素
  - –デンマーク語の aa
  - –スペイン語の ch
- en_USロケール:
  - a < a a < å < ?
- da_DKロケール:
  - a < å < aa < ?
  - → da_DKロケールで [å-?] は aaにマッチ

既存の実装(glibc-2.2.x)の限界

- 照合要素の扱い
- マルチバイトキャラクタ処理速度
- 実装のバグ
- 32KBの制限
- NFAによる制限

glibc-2.2.x vs. glibc-2.3.x(1)

照合要素の扱い

- glibc-2.2.x
  - ○: `[[.a.]-b]`
  - ×: `[a-[.b.]]`
  - ×:`[[=å=]]`, `aa`
  - ×:`[å-?]`, `aa`
- glibc-2.3.x
  - 複数文字照合要素も含め正しく扱える

glibc-2.2.x vs. glibc-2.3.x(2)

マルチバイトキャラクタ処理速度

- glibc-2.2.x
  - mbrtowc のオーバーヘッド
  - 構造上の問題
    - byte指向の処理をad-hoc に wide char 化
- glibc-2.3.x
  - シングルバイトキャラクタの処理と比べ、それほど遅くならない

glibc-2.2.x vs. glibc-2.3.x(3)

実装のバグ

- glibc-2.2.x
  - 10年以上たつにも関わらず枯れていない
  - 度重なる機能拡張が原因?

e.g. `a(b*)¥1+`, `a`

→ 無限ループに陥る

glibc-2.2.x vs. glibc-2.3.x(4)

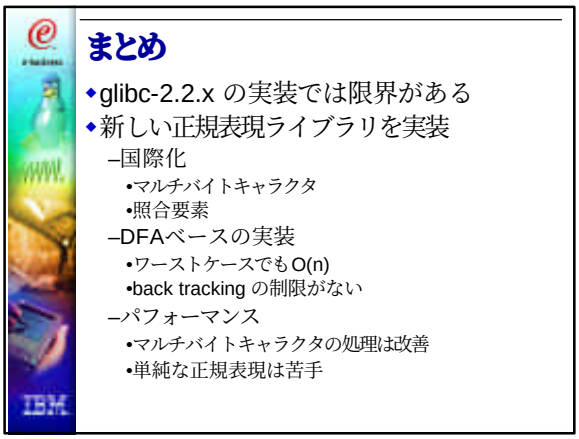
正規表現の長さに32KBの制限

- glibc-2.2.x
  - 2byteの符号付整数でハードコード
- glibc-2.3.x
  - int 型による制限
    32bit 環境では 2GB?

glibc-2.2.x vs. glibc-2.3.x(5)

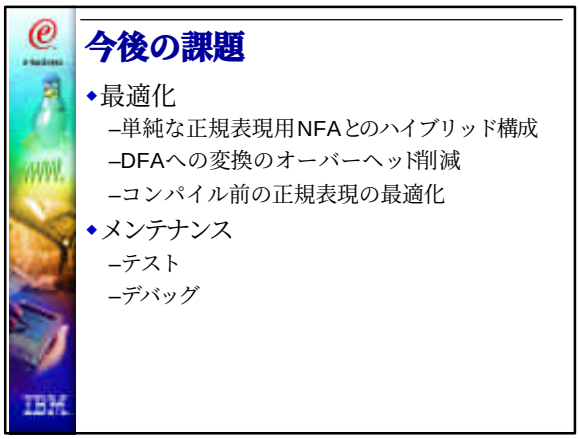
NFAによる限界

- 既存の実装 → NFA
  - back tracking による制限
  - ワーストケースで $O(m^n)$
- 本実装 → DFA
  - back tracking による制限を持たない
  - ワーストケースで O(n)
  - ただし、ただの文字列のような単純な正規表現が苦手



まとめ

- glibc-2.2.x の実装では限界がある
- 新しい正規表現ライブラリを実装
  - –国際化
    - •マルチバイトキャラクタ
    - •照合要素
  - –DFAベースの実装
    - •ワーストケースでもO(n)
    - •back tracking の制限がない
  - –パフォーマンス
    - •マルチバイトキャラクタの処理は改善
    - •単純な正規表現は苦手



今後の課題

- 最適化
  - –単純な正規表現用NFAとのハイブリッド構成
  - –DFAへの変換のオーバーヘッド削減
  - –コンパイル前の正規表現の最適化
- メンテナンス
  - –テスト
  - –デバッグ