# Refsort/Ruby 使用の手引き

**Refsort/Ruby v2.51, IOC List v7.2 準拠**

俊（とし） * †

**2017 年 5 月 5 日**

* toshi.otaguro@nifty.com

† http://griffin.cocolog-nifty.com/lakesidediary/

# 目次

## Refsort/Ruby 使用ライセンス条件

この Ruby スクリプトは以下の条件に従うフリーウェアです．

1. このスクリプトの著作権は原作者が保持します．
2. 誰でもこのスクリプトを自由に使用したりコピーしたり改変することが出来ます．このスクリプトや改変したスクリプトを第 3 者に配布する場合には，必ずスクリプトのソースファイルをそのままの形で配布し，同時に解説書を付属させてください．
3. スクリプト本体である `refsort.rb` と，本解説書 Users Guide（版や書式を特定しない総称）の内容を改変した場合には，必ずファイル名をそれとわかるように変更し，その旨をファイル中に明記し，原作者に知らせたうえで配布してください．
4. 辞書ファイルである `jpblist_v70.ref`[*1]，`jplant054.ref`[*2]，`wbird_sa091a.ref`[*3]，`ioclist_v72.ref`[*4]，`ioclist_v72j.ref`[*5] など（共にパッチレベルやエンコーディングを特定しない総称）は Refsort/Ruby には含まれない独立した著作物です．こちらもコピー，改変，配布は自由ですが，改変したものを配布する場合にはファイル名を他の名前に変え，オリジナルの辞書ファイルには変更を加えないでください．
5. このスクリプトを動作させるために必要な Ruby そのものについては，Ruby のライセンス条件に従ってください．

## 無保証，ならびに使用上の注意点

- このスクリプトの最低限の動作確認は行っていますが，それでも仕様と異なる動作をする可能性があります．お使いになるときはその点を十分理解した上で，システムや重要なファイルのバックアップなどを取った上でお使いください．このスクリプトが原因でシステムが暴走したり，ファイルが破壊されたり，他のいかなる損害が生じたとしても，原作者はその責を問われないものとします．また，原作者はこのスクリプトの今後のバグの修正，改良，配布の義務を負いません．
- この解説書は，Refsort/Ruby v2.51 (Apr-30-2017) に準拠して作成されています．

---

[*1] 日本鳥学会による日本鳥類目録改訂第 7 版を辞書ファイルとして編集したもの
[*2] 日本の種子植物を新エングラー順に並べて辞書ファイルとしたもの
[*3] Sibley-Ahlquist 分類に基づく世界鳥類リストを辞書ファイルとしたもの
[*4] IOC 鳥類リスト v7.2 をそのまま辞書ファイルとしたもの
[*5] IOC 鳥類リスト v7.2 に和名を追加して辞書ファイルとしたもの

- このスクリプトは以下の環境で動作する事が確認されています．
  - Windows 10 Pro version 1703 (build 15063.250) 64bit 日本語版,
    ruby 2.4.1p111 (2017-03-22 revision 58053) [x64-mingw64]
  - Ubuntu 16.04 LTS 日本語 Remix 版 (GNU/Linux 4.4.0-75-generic x64),
    ruby 2.4.1p111 (2017-03-22 revision 58053) [x86_64-linux]
- この手引きの中では，バックスラッシュ記号 '\' が用いられていますが，日本語環境では円記号 '¥' に読み替えてください．
- PC プラットフォームでは，MSWin32 版の Ruby 1.9.1 以降が動く環境であれば動作が期待できます．これこれの環境でこういう不具合があったという報告をしていただければ今後の改善の参考とさせていただきたいと思います．また，バグ報告以外で改善の要望があれば，メールをお寄せください．また下記 URL のホームページ上の掲示板でもご質問，ご要望などを受け付けておりますので，ご利用ください．
- 本スクリプトの転載は自由ですが，必ず事前にメールをお送り下さい．

## Refsort/Ruby の縁起

Refsort は最初は MS-DOS 上で動作する**辞書参照型ソーティング・フィルタ**でした．誕生したのは 1991 年のことです．名前 Refsort の由来は "Reference Sort" の短縮形で，辞書ファイルを参照して並べ替えの順序を決定するソーティング・フィルタという程度の意味です．MS-DOS 上の C コンパイラであった MS-C 6.0 で実装され[*6]，実行形式 `REFSORT.EXE` のサイズはわずか 36k バイトでした．

当時 NIFTY-Serve の FBIRD フォーラム[*7]で公開され，鳥や植物の種名を分類学上の順番に並べ替えるプログラムとして利用されました．同時に，日本鳥類目録改訂第 6 版を辞書ファイル化したものも提供しました．これが最も最初の辞書ファイルです．

あれから 20 年以上が過ぎ PC の環境も大きく変わりました．Windows 上で手軽に Refsort を走らせるため，Ruby というスクリプト言語で実装し直しました．これが Refsort/Ruby です．最初の実装はかなり古く，実は 2001 年のことです．その後何度かの改訂を経たのち，しばらく開発を中断していたのですが，Ruby 本体が 1.9 系列に改版されて，M17N[*8]に基づくエンコーディン

[*6] その後の転職や 2 度の引越しの過程でソースファイルを紛失してしまったのが心残りです．

[*7] 自然愛好家が多く集まり，野鳥を中心にした話題を交換していました．野鳥観察のフィールドノートが多く公開され，またオフライン・ミーティングとしての探鳥会も開催されて大変活発でした．この FBIRD でフィールドノートをアップロードする際の標準書式が定められましたが，自分のフィールドノートを自動的にこの書式に変換してくれる AWK スクリプトを書いたのが，Refsort 開発の最初のきっかけです．

[*8] Multilingualization の最初の M と最後の N をとって名付けられたテキスト処理の多言語化の考え方．

グの取り扱いの根本的な変更が行われたのを機に開発を再開し，さらにその後，この文書の後半で詳しく説明する IOC 鳥類リストを辞書ファイルとして編集するようになってからは，特に精力的に改訂を続けています．

Ruby は**まつもとゆきひろ**さんによるオブジェクト指向のスクリプト言語です．当初から日本語の処理系がしっかりしており，オブジェクト指向であるために自然なプログラミングが可能で習得が楽であること，テキスト処理に必須の正規表現[*9]は Perl と同じくらい強力などの特徴を持っています．当然 MSWin32 の環境でも使用できます．

こんなわけで `REFSORT.EXE` は Refsort/Ruby として生まれ変わりました．オプションの体系や仕様なども大幅に強化されました．Ruby とともに Refsort をごひいきに．

---

最初の M と最後の N の間に 17 文字あることからこのように呼ばれています．

[*9] 文字列を探すための条件を非常に一般性の高い手法で表現したものですが，初心者にとっては呪文のような記号の列にしか見えず何のことだかわかりません．しかし，学ぶに連れてその汎用性と強力さが身にしみて味わえる，ソフトウェア科学の成果の一つと言って良いと思います．Refsort/Ruby の内部では頻繁に使用しており，これなくしては Refsort/Ruby は全く成り立ちません．特に後述するフィールドの分割や別名の処理では大活躍してくれています．

# 1　はじめに

## 1.1　Refsort/Ruby は何ができるのか？

Refsort は辞書参照型のソーティング・フィルタです．Windows には `SORT.EXE` という名前のフィルタがあり，これで通常のソーティング[*10]には十分な機能を持っていますし，Unix 系の `sort` になると更に機能が増えて，オプションの使い方を理解するためにはマニュアルをよく読まなければならないほどです．しかしこれらのフィルタでは，並べ替えの順序は文字コードや数字の大小の順番に固定されており，それでは都合の悪い場合があります．

例えば，フィールドノートの鳥や植物のリストの整理がそうです．探鳥会や植物観察会などではフィールドノートに鳥や植物の名前を記録していきますが，帰宅後にそれを整理するときには，鳥や植物の名前を分類学上の順番に並べ替えてリストにするのが慣習となっています．ところがこれはなかなか厄介な仕事で，鳥のように全体の種類が少ないものでも慣れるまでは図鑑と首っ引きで結構大変ですし，植物のように全体が数千種以上にもなるとこれはもう手仕事では無理です．

実は，今日データの整理や統計処理などによく使われている表計算ソフトの代表格である Microsoft Excel には**ユーザー設定リスト**という機能があり，ユーザーが登録した任意のリストに書かれた順番でデータを並べ替えることができます．しかし，この機能はそれほど数が多くないリストを想定しているようで，せいぜい 200-300 個の要素からなるリストしか登録することができません．太陽系の惑星や衛星の名前とか，星座の名前とか，1 クラスの生徒全員の名前程度であれば十分に用を果たすのですが，生物種の分類となると種類の数が膨大なので，この機能では対処できません．何しろ日本の鳥類だけでも亜種を含めると 1,000 種を超えます．

そこで Refsort が登場します．Refsort は並べ替えの順序を定義した辞書ファイルをあらかじめ用意しておき，入力されたテキスト行をその辞書に記載されている通りの順序に並べ替えてくれるフィルタです．辞書ファイルは単純なテキストファイルですから，自分で自由に作る事が出来ます．分類対象は鳥類でも植物でも哺乳類でも細菌でも何でもかまいません．これからの

---

[*10] ソーティングには実に多様なアルゴリズムがあるのですが，Refsort のアイデアはこれらアルゴリズムとは独立しています．アイデアはごく単純で，辞書ファイル中に定義されている項目とその行番号の組を予め覚えておきます．次にソーティングの対象となる入力リストのそれぞれの文字列を辞書ファイルの中で探し，それが見つかれば覚えておいた行番号をその文字列に付与し，その順序に基づいて入力リストをソートする（それがどのようなアルゴリズムに基づくかは関知しないで）というものです．Microsoft Excel の**ユーザー設定リスト**も同じアイデアを使っていると推測しています．ちなみに，`refsort.rb` の中でソーティングを行なっているのは，`Array#sort` というメソッドですが，これがどういうアルゴリズムを使っているのかを知る必要はありません．おそらく Quick Sort だろうとは思いますが．

解説では，主として日本産鳥類の辞書ファイル jpblist_v70p2w.ref と，日本の種子植物のリスト jplant054w.ref を例にとって説明していきます．

例えば，ある日の探鳥会でフィールドノートに記録した鳥を，出現順のまま，並べ替えたりせずに次のようなテキストファイルにします．半角記号 # はそれ以降がコメントであることを示します．

tamagawa.txt

```
# ----- tamagawa.txt -----
ツグミ                  # 雑木林の落ち葉の上
スズメ                  # 民家の庭
ジョウビタキ            # ♂土手の薮
マガモ                  # 約50羽
カルガモ                # 100羽以上
ヒドリガモ              # 約50羽
オナガガモ              # 約100羽
ハシビロガモ            # ♂♀
セグロセキレイ
トビ                    # 河原の上空
アオジ                  # 川の土手の薮の中
アトリ                  # あまりよく見えなかった
ユリカモメ              # 約20羽
カイツブリ              # 堰堤の上流部
モズ                    # ♂の高鳴き
```

リストの順番は任意です．ファイル名は例えば tamagawa.txt[*11] とでもしておきましょう．これを Refsort というフィルタに通してやります．それには，Windows 環境であればコマンドコンソールを起動し，次のようにコマンドを打ち込みます．ここでは，refsort.rb, jpblist_v70p2w.ref, tamagawa.txt は全てカレント・ディレクトリに置いてあると仮定しています．別のディレクトリ（フォルダ）に置いてある場合にはパス名を含めて適切に指定してください．

```
ruby refsort.rb -f jpblist_v70p2w.ref -nr2 tamagawa.txt
```

すると，あーら不思議，画面には次のようなメッセージとリストが表示されます．

```
!R 52: -R "," redefined
jpblist_v70p2w.ref: 1145 records
tamagawa.txt: 15 records
tamagawa.txt: 15 identified records
tamagawa.txt: 15 unique records
```

[*11] 東京の西郊を流れる多摩川のことです．開発され尽された東京の西郊にあって，この川沿いだけはかろうじて自然が残っている，そういう風景が広がっています．私にとって多摩川が野鳥観察の最初の学校でした．

```
 1 ヒドリガモ              # 約50羽
 2 マガモ                  # 約50羽
 3 カルガモ                # 100羽以上
 4 ハシビロガモ            # ♂♀
 5 オナガガモ              # 約100羽
 6 カイツブリ              # 堰堤の上流部
 7 ユリカモメ              # 約20羽
 8 トビ                    # 河原の上空
 9 モズ                    # ♂の高鳴き
10 ツグミ                  # 雑木林の落ち葉の上
11 ジョウビタキ            # ♂土手の薮
12 スズメ                  # 民家の庭
13 セグロセキレイ
14 アトリ                  # あまりよく見えなかった
15 アオジ                  # 川の土手の薮の中
```

出力されたリストはちゃんと分類学上の目と科の順序に従って並べ替えられており，親切なことに番号まで振ってあります．このように正しく並べ替えられるのは，辞書ファイルである jpblist_v70p2w.ref に日本鳥類目録のリストが書かれているからなのです．画面で見るだけではなく，ファイルに書き出したいのであれば画面出力を適当なファイルにリダイレクトします．

```
ruby refsort.rb -f jpblist_v70p2w.ref -nr2 tamagawa.txt \
    > tamagawa2.txt
```

これで上記のリストが tamagawa2.txt というファイルになりました．画面に出力された部分のうち，jpblist_v70p2w.ref: や tamagawa.txt: で始まる数行は Refsort からの標準エラー出力（コンソール画面）への参考メッセージで，そのままではファイルにリダイレクトされませんので安心してください．こうしておけばすぐに印刷したり，エディタ，表計算ソフトやデータベース・ソフトなどで読み込む事も簡単に出来ます．Web の掲示板にアップロードするときにも，種名の順番の確認で時間を費やす必要がありません．

## 1.2 埋め込みマイルストーン

Refsort/Ruby には**埋め込みマイルストーン**という機能が用意されています．これは辞書ファイルと二人三脚で機能するものですが，辞書ファイル jpblist_v70p2w.ref にはそのための仕掛けが施されています．どういうことかというと，この辞書ファイルには鳥類の目や科や属の名前がコメント行として書かれているのですが，その書き方にはある規則が設けてあります．これらコメントとして埋め込まれたマイルストーンを m オプションによって出力することができます．複数のオプションを併記する場合には mn のようにオプション文字を続けて書くことができます．

```
ruby refsort.rb -f jpblist_v70p2w.ref -mn -r2 tamagawa.txt
```

```
!R 52: -R "," redefined
jpblist_v70p2w.ref: 1145 records
tamagawa.txt: 15 records
tamagawa.txt: 15 identified records
tamagawa.txt: 15 unique records
[ANSERIFORMES; カモ目]
[Anatidae; カモ科]
[Anas; マガモ属]     # "Linnaeus"
     1 ヒドリガモ           # 約50羽
     2 マガモ               # 約50羽
     3 カルガモ             # 100羽以上
     4 ハシビロガモ         # ♂♀
     5 オナガガモ           # 約100羽
[PODICIPEDIFORMES; カイツブリ目]
[Podicipedidae; カイツブリ科]
[Tachybaptus; カイツブリ属] # "Reichenbach"
     6 カイツブリ           # 堰堤の上流部
[CHARADRIIFORMES; チドリ目]
[Laridae; カモメ科]
[Larus; カモメ属]     # "Linnaeus"
     7 ユリカモメ           # 約20羽
[ACCIPITRIFORMES; タカ目]
[Accipitridae; タカ科]
[Milvus; トビ属]      # "Lacepede"
     8 トビ                 # 河原の上空
    ...以下省略
```

なんと！その鳥が属する目や科の名前も出力されていますね！分類に慣れないうちは`m`オプションを使うと勉強になりそうです．さてお役に立ちそうですか？詳しくは以下の使用方法をよく読んでからお使いください．

# 2 インストール方法

## 2.1 Ruby のインストール

まず Ruby 1.9.1 以降のインストールが必要です．公式の Web サイトは以下の通りです．

http://www.ruby-lang.org/ja/

ここで最新版のチェックなどを行ったうえでダウンロードしてください．特に Windows 環境の人は，MSWin32 用にコンパイルされた Ruby[*12]が，

[*12] 従来は 32bit 版しか無かったのですが，現在では 64bit 版も提供されています．お使いの Windows が 32bit 版の場合は Ruby も 32bit 版を使うのですが，Windows が 64bit 版の場合は Ruby の 32bit 版でも 64bit 版でも使うことができます．

http://rubyinstaller.org/
https://github.com/oneclick/rubyinstaller2/releases/
http://www.artonx.org/data/asr/
http://ruby.morphball.net/rumix/

などにありますので，そこからダウンロードすると良いでしょう．詳しくは上記 Ruby の公式サイトで調べてみてください．いずれの場合も Ruby そのもののインストールは自力で[*13]行っていただく必要があります．ダウンロードしたファイルの中にインストール方法が詳しく書いてあるはずですから，よくお読みの上でインストール作業を行ってください．

MSWin32 版のインストールは，付属のインストーラを起動するか，インストーラが無い場合には任意のディレクトリで展開するだけです．例えば，`C:\ruby` というディレクトリを作り，そこで展開するとその下に `bin`，`doc`，`include`，`lib`，`share` という 5 つのディレクトリが作られ，これらの中に多数のファイルが展開されます．

Ruby の本体は `bin` の中にある `ruby.exe` です．従って今の場合には `C:\ruby\bin` を環境変数 `PATH` の値に含めておく必要があります．環境変数の変更について詳細は省略しますが，`PATH` の設定を忘れがちなのでご用心ください．インストーラがついている場合には，`PATH` は自動で追加されることがほとんどだと思います．ちなみに，Rubyinstaller や Rubyinstaller2 では，インストールに成功すると，Windows のスタートメニュの中に Rubyintaller のフォルダが作られ，その中にコンソールを起動するアイコンが作られています．このアイコンから起動されるコンソール内では，`ruby.exe` への `PATH` が自動的に付加されてますので，どのディレクトリであっても，`ruby` と打ち込めば Ruby が実行されます．

インストールが終了したら コマンドコンソールから，

```
ruby --version
```

などとして正しくインストールされたかどうか確認してください．

[*13] Ruby は汎用の処理系として構築されているため，正式にはソースファイルの提供しかありません．従って Unix 系 OS では，適当なパッケージ管理システム（Debian 系では apt という有名なパッケージ管理システムがあります．また rbenv や RVM という Ruby のバージョン管理システムもあります）に登録されている処理系をインストールするか，あるいはソースファイルから自力で処理系をビルドする必要がありますが，いずれにせよさほど手間はかかりません．一方，Windows 環境では自力ビルドを行うことは一般的ではなく，第三者の有志がビルドしてくれたシステムをインストールするのが普通です．それが上記の 4 つの Web サイトにあるのですが，同一の処理系とは言い切れない面があります．

## 2.2 Refsort (refsort.rb) のインストール

インストールとは言っても，refsort.rb は単一の Ruby スクリプトにすぎません．Refsort を用いた作業を行いたいフォルダにコピーするか，Ruby スクリプトをまとめて置くと決めたフォルダーにコピーする，などとすれば良いのです．こちらも例えば，

```
ruby refsort.rb --version
```

と打ち込むと，

```
refsort.rb v2.51, Apr-30-2017
```

と refsort.rb のバージョンが返ってくれば正しくインストールされています．

## 2.3 辞書ファイルのインストール

辞書ファイルが無くては Refsort/Ruby は十分には機能しません．現在，以下のような辞書ファイル（エンコーディングやパッチレベルを特定しない総称）が使用可能です．

| ファイル名 | 内容 |
|---|---|
| jpblist_v70.ref | 日本鳥類目録 改定第 7 版 |
| jplant054.ref | 日本の種子植物リスト |
| wbird_sa091a.ref | Sibley-Ahlquist 分類 世界鳥類リスト |
| ioclist_v72.ref | IOC World Bird List v6.3 オリジナル英名版 |
| ioclist_v72j.ref | IOC World Bird List v6.3 和名追加版 |

これらのファイルもダウンロードして，ソート作業を行うディレクトリに置くか，辞書ファイルを置くと決めたディレクトリにコピーし，パス名を含めて f オプションで指定してください．

# 3 フィールドとその分割

## 3.1 レコードとフィールド

フィールドという概念はソーティングにとっては大変重要です．Refsort/Ruby では，フィールドの区切り方を，できる限り私たちの思考パターンに沿った自然な形とするように工夫しています

が，それについても最初に説明しておきます．すこし難しい部分もありますが，大変重要なので，できるだけ理解してください．

Refsort/Ruby は，並べ替えの基準となる辞書ファイルも，並べ替えの対象となる入力も，一行（**レコード**）ずつ読み込み処理していきます．一行を読み込むとそれを行頭から調べていき，あらかじめ定義された区切り記号に出会うたびにひと塊の文字列である**フィールド**に分割していきます．フィールドには 0 から始まる番号が付けられます．デフォルトの状態では空白文字（半角の空白とタブ）が区切り記号なので，鳥の和名と英名を並べて書いた

```
マガモ Mallard
```

という行は，

```
第 0 フィールド = マガモ
第 1 フィールド = Mallard
```

という具合に分割されます．しかしマガモの学名 Anas platyrhynchos を和名と英名の間に書き加えた行

```
マガモ Anas platyrhynchos Mallard
```

に対しては，空白文字が区切り記号であるために，学名は以下のように二つのフィールドに分割されてしまいます．

```
第 0 フィールド = マガモ
第 1 フィールド = Anas
第 2 フィールド = platyrhynchos
第 3 フィールド = Mallard
```

種のレベルに限れば学名は常に二つの単語から成るという規則があるから良いのですが，亜種や変種のレベルに降りた場合には学名もいくつの単語から成るのか予測できませんし，そもそも英名を構成する単語の数は不定なので，全体でフィールドがいくつになるのか予測できません．例えば，カルガモの英名は Eastern Spot-billed Duck で 3 単語から成り，コガモは Eurasian Teal と 2 単語から成ります．空白文字を区切り記号にするとこのような場合には不都合ですから，別の記号，例えばカンマを区切り記号にして，

```
マガモ,Anas platyrhynchos,Mallard
```

とすれば，これは確実に和名，学名，英名をフィールドに分割できます．

```
第 0 フィールド = マガモ
第 1 フィールド = Anas platyrhynchos
第 2 フィールド = Mallard
```

あるいは，区切り記号はデフォルトの空白文字のままで，空白文字を含む学名全体を二重引用符で囲んでも同じことができます．

```
マガモ "Anas platyrhynchos" Mallard
```

これらの方式を両立させることもできます．また，区切り記号に隣接する任意の個数の空白文字は無視されますので，

```
マガモ,"Anas platyrhynchos",Mallard
マガモ,    "Anas platyrhynchos"    ,  Mallard
```

のように書いても等価です．

## 3.2 フィールド分割規則

このフィールド分割の規則をきちんと書くと，以下のようになります．

1. コメント記号 # 以降を取り除く．ただし行頭から #! で始まるコメント行は，埋め込みマイルストーンや埋め込みオプションとして特別に扱う．（後述）
2. 空白文字からのみ成る行や改行だけの行は無視する．
3. あらかじめ定義された文字の集合の要素（区切り記号）を探し，以下の規則でフィールドに分解する．
   （a）デフォルトの区切り記号は空白文字（半角の空白とタブ）．区切り記号が陽に定義されている場合には，その中に予約記号の !, ?, #, |, \, " が含まれていないことを確認し，予約記号が含まれている場合にはエラーメッセージを表示して処理を中止する．区切り記号に空白文字が含まれている場合にはそれを取り除き，その旨の警告を出力して処理を続行する．
   （b）行頭，行末の任意の個数の連続した空白文字と，区切り記号に隣接する任意の個数の連続した空白文字は取り除き，区切り記号によって区切られたフィールドに分割する．
   （c）二重引用符で囲まれた文字列の中では，区切り記号は普通の文字として扱う．
   （d）二重引用符で囲まれた文字列の中では，二重引用符自身はバックスラッシュ（日本語環境では円記号 ¥[*14]）でエスケープする (\"または ¥") か，二重引用符を 2 個連続して書く ("") ことで表現することが出来る[*15]．またタブは \t で表現できる．

[*14] このガイドブックの 4 ページ “**無保証，ならびに使用上の注意点**” にも書かれているとおり，本マニュアル中では，円記号はすべてバックスラッシュで表現されていますので，日本語環境ではこれらを円記号に読み替えてください．

[*15] Refsort は二通りのエスケープ方法を提供しています．これはカンマ区切り書式 CSV では伝統的に 2 個連続した二重引用符が使われ，また Ruby など一部の処理系では二重引用符をバックスラッシュでエスケープすることがデフォルトとなっているためです．

## 3.3 フィールド分割の例

例をあげます．各行のコメント記号 #=> の右側は分割されたフィールドの内容を示します．

区切り記号を指定しない場合 (空白文字が区切り記号)

```
a  b  c                          #=> "a", "b", "c"
 "a" " b c " "d"                 #=> "a", " b c ", "d"
   a  " b " c                    #=> "a", " b ", "c"
a "b, c" d                       #=> "a", "b, c", "d"
a "b \"c\" d" "e \"f\""          #=> "a", "b "c" d", "e "f""
a "b ""c"" d" "e ""f"""          #=> "a", "b "c" d", "e "f""
```

カンマとコロンを区切り記号に指定した場合

```
x, y : z                         #=> "x", "y", "z"
a, "b : c", d                    #=> "a", "b : c", "d"
  a: "bc, d", e                  #=> "a", "bc, d", "e"
a, "b \"c\" d", "e \"f\""        #=> "a", "b "c" d", "e "f""
a, "b ""c"" d", "e ""f\""        #=> "a", "b "c" d", "e "f""
```

日本鳥類目録に準拠した日本の鳥類の辞書 jpblist_v70p2.ref や日本の種子植物の辞書 jplant054.ref，Sibley-Ahlquist 分類の世界の鳥のリスト wbird_sa09a.ref，IOC World Bird List に準拠した鳥類辞書 ioclist_v72.ref などはいずれもカンマを区切り記号にしています．これにより，不定個の単語からなる英名も正しく扱うことができますが，上の例のように二重引用符で囲むやり方でも同じことが実現できます．

# 4 辞書書式

## 4.1 基本要件，コメントなど

辞書ファイルは以下の規則に則って作成されなければなりません．

1. テキストファイルであること．全角半角文字は混在してよい．エンコーディングも US-ASCII，Shift_JIS，Windows-31J，EUC，UTF-8 など通常使われるものであれば何を使ってもよい．一行の最大長や行数は Ruby の制限に従うが，実用上は特に制限は無いと考えてよい．空白行やコメント行を取り除いた後の実質名（後述）の最大数も制限は無い．
2. 半角の記号 # はコメント記号として予約されており，その文字以降はコメントとして無視される．

3. 空白行や，コメントのみの行は無視される．

## 4.2 エンコーディング

1. 辞書ファイルの第 1 行目で，行頭が `#!E` で始まり，`#!E` の直後に 0 個以上の任意の文字を置き，さらにその後に `coding:` という文字列を置き，さらに直後に 0 個以上の空白文字（半角空白またはタブ）を置いたのちに現れる最初の単語，すなわち正規表現（脚注 *9 をご覧ください）では

   ```
   /^#!E.+?coding[:=]\s*([\w.-]+)/
   ```

   の () 内にマッチする文字列，例えば，`Windows-31J` や `UTF-8` が，その辞書ファイルのエンコーディングとみなされる．このコメント行が無い辞書ファイルの場合には，辞書ファイルのエンコーディングはそのシステムで最も自然な言語環境 Locale に従う．例えば，Windows 7 や Windows 8.1 の場合には，`Windows-31J` というエンコーディングが適用される．MS-DOS 時代の `Shift_JIS` ではないことに注意．[*16]ちなみに日本産鳥類の辞書ファイル `jpblist_v70p2w.ref` では，その第一行は次の通り．

   ```
   #!E -*- coding: Windows-31J -*-
   ```

2. 辞書ファイルのエンコーディングと入力のエンコーディングは**同一でなければならない**．同一でない場合には，Refsort はエラーメッセージを出して直ちに終了する．

## 4.3 区切り記号

1. 区切り記号という特殊な文字の集合をコマンドラインの `R` オプションによって定義することができる．定義しない場合は空白文字（半角の空白とタブ）が区切り記号として指定されたことと等価である．通常の半角文字や全角文字であればほとんどのものは区切り記号として定義することができる．例えば，`-R ","` というオプションを指定すれば，半角のカンマを区切り記号として定義することができる．タブを指定する場合には `-R "\t"` とする．
2. 区切り記号によって区切られた文字列をフィールドと呼ぶ．コマンドラインの `r` オプションによって順序と位置を指定された複数のフィールドの並び（区切り記号を含まない）のことを**実質名**と呼ぶ．この実質名の出現順序が並べ替えの基準となる．例えば，`-r 1-2` と指定

[*16] 現在でも Windows-31J が Shift_JIS と同一であると思っている人が多く，フリーソフトのエディタにもそのような設定が見られるのですが，厳密には異なります．MS-DOS から Windows に移行するときに Microsoft はエンコーディングも改訂したのですが，機種依存文字という厄介な問題が発生しました．詳しくは Wikipedia の記事をご覧ください．Windows-31J は別名 CP932 (Code Page 932) としても知られています．

すると，第 1 フィールドから第 2 フィールドの並びを実質名とすると指定したことになる．フィールドの番号は 0 から始まる．

3. コマンドラインで R オプションを指定しないデフォルトの状態では，区切り記号は空白文字（半角の空白とタブ）であるが，連続した空白文字は 1 個の空白文字と等価に扱われ，行頭，行末の連続した空白文字は無視される．
4. これと同様に，R オプションによって区切り記号をユーザーが指定した場合にも，非空白文字の区切り記号の前後にある複数の連続した空白文字と，行頭，行末の連続した空白文字は無視される．
5. R オプションによって区切り記号をユーザーが指定した場合，複数の連続した区切り記号は複数の空のフィールドを作る．
6. 辞書ファイルの中の `#!R` で始まる行は**埋め込みオプション**の一種で，この場合には，それ以降の行には，`#!R` の後の 0 個以上の空白文字に続く二重引用符で囲まれた文字セットを区切り記号として使用するよう指定したことになる．この指定は上記の R オプションによるものを取り消して，新たに区切り記号を指定する効力を持つ．従って，

   ```
   #!R ","
   ```

   という行があれば，辞書ファイルのそれ以降の行には，フィールドの区切り記号として半角のカンマ `","` が指定されたことになる．この埋め込みオプションは複数の行に置くことができるので，一つの辞書ファイルの中の途中から，区切り記号を変更することができる．これは，複数の辞書ファイルを合体させて一つの辞書ファイルとして使用したい場合などに便利である．
7. 区切り記号をデフォルトの状態 (空白文字) に戻したいときには，空の文字列 `""` を指定すればよい．

辞書ファイルの作者は，`#!R` オプションを埋め込むことによって，利用者が区切り記号を意識せずに使用できるように配慮しておくことが望ましい．

**■アマサギの例**　アマサギ[*17]の和名，英名，学名を記述する場合の例を以下に示す．デフォルトでは，

```
アマサギ  "Eastern Cattle Egret"  "Bubulcus coromandus"
```

[*17] 以前はアマサギの学名は Bubulcus ibis で，英名は Cattle Egret とされてきました．しかし最近では B. ibis の一亜種であり日本でアマサギとして知られる亜種アマサギの B. ibis coromandus が種に格上げされ，学名が B. coromandus，英名が Eastern Cattle Egret となったため，このような表記としました．ちなみに，B. ibis coromandus が独立して出て行った後の残りの B. ibis の英名が Western Cattle Egret に変更されたため，和名もそれにならって暫定的に**ニシアマサギ**としています．このような変更が IOC List では頻繁に発生しており，和名をどう再定義するかは非常に厄介な問題です．

と書けば，

```
第 0 フィールド = アマサギ
第 1 フィールド = Eastern Cattle Egret
第 2 フィールド = Bubulcus coromandus
```

と分割される．一方 コマンドラインで -R "," と指定した場合には，カンマが区切り記号なので，

```
アマサギ， Eastern Cattle Egret， Bubulcus coromandus
```

と書いても，

```
アマサギ, "Eastern Cattle Egret", "Bubulcus coromandus"
```

と書いても，

```
第 0 フィールド = アマサギ
第 1 フィールド = Eastern Cattle Egret
第 2 フィールド = Bubulcus coromandus
```

と分割される．

## 4.4 フィールド指定

どのフィールドを並べ替えの基準にするのか，以下のように指定します．

1. 辞書ファイルの中の #!r で始まる行も埋め込みオプションの一種で，この場合には辞書ファイルのフィールド番号の指定を途中から変更することができる．これもコマンドラインの r オプションによる指定を取り消し，新たにフィールド番号を指定する．例えば，コマンドラインで r オプションを指定しない場合には，-r 0 を指定したのと等価であるが，これを途中から変更したい場合には，

   ```
   #!r 1
   ```

   などという行を書いておけば，この行以降は第 1 フィールドを指定したことになる．フィールド番号を指定しないもの #!r は #!r 0 と等価である．また，#!r abc などとフィールド番号の指定として意味の無いものを書いても，それは単に無視される．
2. このオプションは #!R オプションと同様に辞書ファイル中の複数の行におくことが出来るので，複数の辞書ファイルを合体して一つの辞書ファイルとして使用したい場合などに便利である．
3. 注意すべきは，一度指定されたフィールド番号を埋め込みコメントで変更することはできるが，フィールドの数を変更することはできないことである．特に間違いやすいのは，コマンドラインで何も指定しない場合で，これはデフォルトで -r 0 と指定したのと等価であ

り，対象フィールの数は 1 となっている．その後辞書ファイルで `-r 1-2` などと再指定しようとしても，フィールド数が合わないというエラーとなり処理は中断される．これは複数のフィールドを持つレコードを並べ替える際に途中でフィールド数を変えることはできないという原理的な制限であり，Refsort の機能による制限ではない．

4. 上記 `r` オプションの書式には柔軟性がある．すなわち，`-r` の後に空白文字を置き，さらに続けてフィールド番号をカンマで区切って（空白をはさまずに）並べる．連続したフィールド番号を指定するときには，ハイフンでその両端を指定することができる．例えば，

```
-r 0-2,4,5    # -r 0,1,2,4,5 と同じ
-r 3-1,0,2    # -r 3,2,1,0,2 と同じ
```

これにより，辞書の中の任意で不連続なフィールドを選択して並べ替えの基準とすることが可能となる．

## 4.5 別名

辞書ファイルの各フィールドには複数の**別名**あるいは *synonym* を並べて書くことができる．Refsort/Ruby はこれらを同一のものと見なす．例えば日本の種子植物の辞書ファイル `jplant054w.ref` には以下のような行がある．

```
アオモリトドマツ|オオシラビソ，  Abies mariesii
```

これはアオモリトドマツとオオシラビソが互いに同一の種を表す別名であることを示している．別名を並べるには，区切り記号 '`|`' (半角の縦棒) を用いる．分割された単一のフィールド内に複数の別名を '`|`' で区切って並べた文字列が存在する場合には，区切り記号の両側に複数の空白文字を置いてもそれらは無視される．またフィールドの先頭や末尾に別名記号があっても単に無視される．別名に指定されている文字列は，そのいずれが入力に現れても，互いに同一のものとして扱われ，ソートの順番も同等である．

例えば，フィールド区切り記号がデフォルトの場合には，

```
"a | b" c d
```

と書くと，`a` と `b` とは互いに別名である．フィールド区切り記号が `","` である場合に

```
a | b , c, d
```

と書くのと等価である．

同一レコードの複数のフィールドに別名を指定することもできる．例えば，

```
a | b | c,  p,  u | x
```

と書けば，これは

```
a, p, u
b, p, u
c, p, u
a, p, x
b, p, x
c, p, x
```

という 6 種類の組み合わせを全て同等のものと見なす．

植物分類の世界では，同一の種に対して複数の別名が用いられることが多いので，辞書ファイルに別名を指定することによって別名どうしを矛盾なく扱うことができる．

また分類学の進展により，従来考えられていた分類体系が修正される過渡期には，古い学名と新しい学名が併用されることが多いが，そのような場合にもこの別名を使用できる．

## 4.6 埋め込みマイルストーン

埋め込みマイルストーンも Refsort の大きな特徴の一つです．これについて説明します．

行頭が `#!m` で始まるコメント行は**埋め込みマイルストーン**と呼ばれる特殊なコメント行である．行頭の `#!m` の直後に 0 個以上の空白文字（半角空白またはタブ）を置き，さらにその後に任意の同一の半角文字を 1 個以上並べ，さらにその後に任意の文字列を置き，さらに記号 `#` に続いてコメントを置くことができる．以下に例を示す．

埋め込みマイルストーンの例

```
#!m >[ア ビ 目]
#!m >>[ア ビ 科]
ア ビ
オ オ ハ ム
シ ロ エ リ オ オ ハ ム
ハ シ グ ロ ア ビ
ハ シ ジ ロ ア ビ
#!m >[ミ ズ ナ ギ ド リ 目]
#!m >>[ア ホ ウ ド リ 科]     # 亜 科 以 降 は 省 略
コ ア ホ ウ ド リ
ク ロ ア シ ア ホ ウ ド リ
ア ホ ウ ド リ
```

埋め込みマイルストーンの使用規則は以下のとおり．

1. 行頭の `#!m` の後の 0 個以上の空白文字に続く 1 個以上の同一の半角文字（上の例では ‘>’）の個数には埋め込みマイルストーンの**レベル**という意味がある．上の例ではアビ目とミズナギドリ目のレベルは 1 でありアビ科とアホウドリ科のレベルは 2 である．埋め込みマイル

ストーンオプション m が指定されている場合には，埋め込みマイルストーンはそのレベルとともに記憶され，並べ替えた入力行を出力する際に以下の規則に従って出力される．

2. 入力行の指定された**実質名**が，辞書ファイルのある行の指定された実質名と一致した場合，入力行は辞書ファイルの順に並べ替えられて出力されるが，その出力に先立ち，辞書ファイルでその行の最も直前に出現した全てのレベルの埋め込みマイルストーンのうち，未出力のものがレベルの昇順に出力される．ただし埋め込みマイルストーンの行頭の `#!m`，それに続く空白文字，レベルを表す半角の文字列は取り除かれる．
3. 埋め込みマイルストーンには，レベル記号やコメントの実体のさらに後に，記号 '#' に続けてコメントを書くことができる．このコメントは出力時にはそのまま出力される．ただし例外として，数字付きコメント，例えば #2 があればそれ以降の文字列は全て取り除かれる．また，辞書ファイル読み込み時に自動的に行われる埋め込みマイルストーンの重複チェックでは，コメントを取り除いた部分でのみ重複チェックが行われる．

例えば，入力が下記の場合，

```
アホウドリ
オオハム
アビ
```

これらは辞書ファイルに従って，アビ，オオハム，アホウドリの順に出力されるが，アビに対しては埋め込みマイルストーン [アビ目] と [アビ科] が出力される．オオハムに対しては最も近い埋め込みマイルストーンは全レベル既出力であるので，埋め込みマイルストーンは出力されない．アホウドリに対しては最も近い埋め込みマイルストーンが同様に出力されるので，出力全体は以下のようになる．

```
[アビ目]
[アビ科]
アビ
オオハム
[アホウドリ目]
[アホウドリ科]      # 亜科以降は省略
アホウドリ
```

## 4.7 その他

1. 辞書ファイルは並べ替えの基準となるファイルであるから，その作成には細心の注意を払わなければならない．少なくとも重複した実質名が無いように確認する必要がある．万一辞書ファイル中に重複した実質名があった場合には，Refsort はその行番号と実質名を標準エラー出力（コンソール画面）に出力し，直ちに処理を終了する．

2. Refsort は全角，半角，大文字，小文字を全て厳密に区別する．もしも辞書ファイルに全角カタカナでリストが作成されている場合には，入力にも全角カタカナを使わないと，入力された名前が辞書中に見つからないと言うエラーになる．ただしｃオプションが指定されていれば，半角の英文字の大文字/小文字は区別しない．このｃオプションが指定されていない場合には，半角英文字の大文字/小文字を区別するので辞書ファイルと入力ファイルの整合性に注意すること．また単語間の空白の数も全て厳密に同一でなければならないので，何かのきっかけで単語間に二つ以上の空白文字があった場合には，入力もそのようになっていなければ同一のものと見なされない．これを救済するにはｓオプションを使用する．（後述）
3. 辞書ファイルの中でハイフン記号 "-" が使われているにも関わらず，入力ファイルでは何らかの理由でハイフン記号が空白文字に置き換えられている，あるいはその逆が起こる場合がある．例えば鳥類の英名には，Common Paradise-kingfisher と書いたり，Common Paradise Kingfisher と書いたりするものがある．この用法には種々の流派があり一つに特定することはできないため，これを吸収するための機能としてｂオプションを設けた．このオプションを使用すると，辞書ファイルと入力ファイルの両方で，フィールド分割を行った後の実質名において，ハイフン記号 "-" と空白文字 "␣" を同一視する．このオプションは，英文字の大文字/小文字の区別を無視するｃオプションと併用すると，上記のような用法の多様性を吸収できる可能性が高くなる．
4. 実質名が空だった場合（空白文字ではなく空，つまり実質名が無い場合）には，それを記憶格納せず．そのレコードは無かったものとして無視し，警告文を出して処理を続行する．ところで，実質名が空かどうかというのは，単一のフィールドの場合にはすぐに想像できるが，実質名が複数のフィールドから成る場合がある．この場合には，指定された全てのフィールドが空の場合にのみ，実質名は空であると定義している．例えば，

```
#!R ","
#!r 0-2
a,  ,
 , b,
  ,  , c
 ,  ,
```

という各行に対しては，最後のレコードのみが実質名が空と判定され，他のレコードでは空とは判定されないことに注意すること．

# 5　入力書式

## 5.1　基本要件，コメントなど

入力は以下の書式に従って作成されなければなりません．考え方は辞書ファイルとほとんど同じです．ただし入力ファイルには埋め込みマイルストーンや別名を指定することはできません．

1. テキストファイルである事．全角半角文字は混在してよい．一行の最大長は Ruby の制限に従うが実用上は気にしなくて良い．行数も Ruby の制限に従うが，実用上は特に制限は無いと考えてよい．空白行やコメント行を取り除いた後の**実質名**の最大数も制限は無い．
2. 半角の記号 `#` はコメント記号として予約されており，その文字以降はコメントとして処理される．
3. 空白行や，コメントのみの行は無視される．
4. 入力ファイルの第 1 行目で，行頭が `#!E` で始まり，`#!E` の直後に 0 個以上の任意の文字を置き，さらにその後に `coding:` という文字列を置き，さらに直後に 0 個以上の空白文字 (半角空白またはタブ) を置いたのちに現れる最初の単語，例えば，`Windows-31J` や `UTF-8` がその入力ファイルのエンコーディングとみなされる．このコメント行が無い入力ファイルの場合には，入力ファイルのエンコーディングには，そのシステムの最も自然な言語環境 Locale に従うと仮定される．例えば Windows 7 の Locale は `Windows-31J` であり，`Shift_JIS` ではないことに注意．

## 5.2　エンコーディング

入力のエンコーディングも辞書ファイルのエンコーディングと同様の規則が当てはまりますが，エンコーディングを一致させるための注意が必要になります．

1. 入力のエンコーディングは辞書ファイルのエンコーディングと**同一でなければならない**．同一でない場合には Refsort はエラーメッセージを出して直ちに終了する．
2. エンコーディングにはさらに注意が必要である．プラットフォームが Windows（日本語版）の場合は，システムの自然なエンコーディングは Windows-31J なので，辞書ファイルが Windows-31J でエンコードされていれば，標準入力から，すなわちキーボードから直接入力することも可能であるし，Windows-31J でエンコードされているファイルをリダイレクトやパイプで入力することもできる．一方，UTF-8 でエンコードされた辞書ファイルも使うことができるが，この場合に許される入力方法は，UTF-8 でエンコードされた入力ファ

イルをコマンドラインで指定することのみである．しかもこの入力ファイルの第 1 行にはエンコーディングの指定がなければならない．

3. プラットフォームが Linux の場合は Locale が UTF-8 なので，辞書ファイルが UTF-8 でエンコードされている限り，UTF-8 でエンコードされている入力は，コマンドラインで入力ファイルを指定しても，標準入力やリダイレクトで与えてもよい．一方，Windows-31J でエンコードされた辞書ファイルを使うこともできるが，この場合には，Windows-31J でエンコードされ，かつ第 1 行にエンコーディング指定のある入力ファイルをコマンドラインで指定することのみ許される．
4. 出力のエンコーディングは**入力のエンコーディングと同じもの**が用いられる．従って，使用しているプラットフォームの Locale とは異なるエンコーディングのファイルを入力すると，出力にも同じエンコーディングが適用される．出力の改行コードも入力ファイルのそれと同じものが用いられる．

## 5.3 区切り記号

入力の区切り記号については，辞書ファイルの場合と同様の規則が当てはまります．

1. 区切り記号という特殊な文字の集合をコマンドラインの `I` オプションによって定義することができる．デフォルトでは空白文字（半角の空白とタブ）が区切り記号として定義されているが，通常の半角文字であればほとんどのものは区切り記号として定義することができる．例えば，`-I ","` というオプションを指定すれば，半角のカンマを区切り記号として定義することができる．タブは `\t` によって指定できる．
2. 区切り記号によって区切られた文字列を**フィールド**と呼ぶ．コマンドラインの `i` オプションによって順序と位置を指定された複数のフィールドの並び（区切り記号を含まない）のことを**実質名**と呼ぶ．この実質名の出現順序が並べ替えの基準となる．例えば，`-i 1,2` と指定すると，第 2 番目のフィールドから第 3 番目のフィールドの並びを実質名とすると指定したことになる．フィールドの番号は 0 から始まる．
3. コマンドラインの `I` オプションを指定しないデフォルトの状態では，区切り記号は空白文字であるが，連続した空白文字は 1 個の空白文字と等価に扱われ，行頭，行末の連続した空白文字は無視される．
4. これと同様に，`I` オプションによって区切り記号をユーザーが指定した場合にも，非空白文字の区切り記号の前後にある複数の連続した空白文字と，行頭，行末の連続した空白文字は無視される．
5. コマンドラインの `I` オプションによって区切り記号をユーザーが指定した場合，複数の連続した区切り記号は複数の空のフィールドを作る．

6. 入力の中の `#!I` で始まる行は**埋め込みオプション**の一種で，この場合には，`#!I` の後の 0 個以上の空白文字に続く二重引用符で囲まれた文字セットを，それ以降の行に対して区切り記号として使用するよう指定したことになる．この指定は上記のコマンドラインの `I` オプションによるものを取り消して，新たに区切り記号を指定する効力を持つ．従って入力ファイル中に，

   ```
   #!I ","
   ```

   という行があれば，入力のそれ以降の行には，フィールドの区切り記号として半角のカンマ `","` が指定されたことになる．この埋め込みオプションは複数の行に置くことができるので，一つの入力ファイルの中の途中から，区切り記号を変更することができる．これは，複数の入力ファイルを合体させて一つの入力ファイルとして使用したい場合などに便利である．
7. 区切り記号をデフォルトの状態 (空白文字) に戻したいときには，空の文字列 `""` を指定すればよい．

## 5.4 フィールド指定

入力のフィールド指定についても辞書ファイルの場合と同様の規則が適用されます．

1. 入力中の `#!i` で始まる行も埋め込みオプションの一種で，この場合には入力のフィールド番号の指定を途中から変更することができる．これもコマンドラインでの `i` オプションによる指定を取り消し，新たにフィールド番号を指定する．例えば，コマンドラインで `i` オプションを指定しない場合には，`-i 0` を指定したのと等価であるが，これを途中から変更したい場合には，

   ```
   #!i 1
   ```

   などという行を書いておけば，この行以降は第 1 フィールドを指定したことになる．フィールド番号を指定しないもの `#!i` は `#!i 0` と等価である．また，`#!i abc` などとフィールド番号の指定として意味の無いものを書いても，それは単に無視される．
2. このオプションは `#!I` オプションと同様に入力中の複数の行におくことが出来るので，複数の入力ファイルを合体して一つの入力として使用したい場合などに便利である．
3. 注意すべきは，一度指定されたフィールド番号を埋め込みコメントで変更することはできるが，フィールドの数を変更することはできないことである．特に間違いやすいのは，コマンドラインで何も指定しない場合で，これはデフォルトで `-i 0` と指定したのと等価であり，対象フィールの数は 1 となっている．その後辞書ファイルで `-i 1-2` などと再指定しようとしても，フィールド数が合わないというエラーとなり，処理は中断される．これは複数のフィールドを持つレコードを並べ替える際に途中でフィールド数を変えることはできないと

いう原理的な制限であり，Refsort の機能による制限ではない．

4. 上記 i オプションの書式には柔軟性がある．すなわち，-i の後に空白文字を置き，さらに続けてフィールド番号をカンマで区切って（空白をはさまずに）並べる．連続したフィールド番号を指定するときには，ハイフンでその両端を指定することができる．例えば，

```
-i 0-2,4,5    # -i 0,1,2,4,5 と同じ
-i 3-1,0,2    # -i 3,2,1,0,2 と同じ
```

これにより，入力行の中の任意で不連続なフィールドを選択して並べ替えの基準とすることが可能となる．

5. 入力中に重複した実質名があり，かつ d オプションが指定されている場合には，Refsort は警告としてその行番号と実質名を標準エラー出力（コンソール画面）に出力し，処理を続行する．さらに，u オプションを付けない限り，出力結果には重複した実質名を含む入力行が入力された順に隣合った行に出力される．d オプションを指定していなくて重複した実質名があった場合でも同様の出力結果を得るが，警告メッセージは出力されない．

### 5.5 その他

1. タイプミス，全角と半角の混同などで，辞書ファイル中にない実質名を入力として与えた場合には，Refsort は標準エラー出力（コンソール画面）に警告としてその入力の行番号と行内容とを出力して処理を続行する．
2. Refsort は入力行から実質名を取り出し，それを元に並べ替えを行った後で入力行を**入力された内容のまま**出力する．従って入力行中に含まれている行頭行末の区切り記号やコメントなどもそのまま出力される．ただし，u オプションを付けた場合には実質名だけが行頭から出力される．また実質名を含まないただの空白行やコメント行は出力されない．ただし，出力オプション o を指定した場合にはこのルールは適用されない．詳しくは下記第 6 節の “**出力形式**” を参照のこと．

## 6 出力形式

Refsort/Ruby v1.34 以前の版では，出力に関する自由度はほとんど無く，入力行を丸ごと出力するか，u オプションが指定されている場合には実質名のみを出力することしかできませんでした．応用範囲を広げるために v2.00 では大幅な自由度を加えました．そのルールは以下の通りです．

## 6.1 フィールド指定

1. コマンドラインにoオプションが無い場合には，入力行全体をそのまま出力する．oオプションが無くuオプションがある場合には，入力された実質名のみが出力される．
2. コマンドラインにoオプションがある場合は，oに空白文字を経て次の空白文字に出会うまでの文字列が出力フィールドの並びになる．出力フィールドの自由度は高く，‘r’と‘i’の接頭辞で辞書ファイルのフィールドか入力ファイルのフィールドかを区別し，それに接する数字でフィールド番号を指定する．例えば

   ```
   -o i0,r0,i3-1,r2-1
   ```

   のように辞書ファイルのフィールド，入力行のフィールドを混在させて自由な順序で選択することができる．またフィールド全体（コメント部分は除く）を指定する場合には‘ra’や‘ia’という記号を使うことができる．上記の場合には，入力の第0フィールド，辞書の第0フィールド，入力の第3フィールドから第1フィールドまで，辞書の第2フィールドから第1フィールドまでを並べて出力することを意味する．
3. 出力フィールドに辞書ファイルや入力行のコメント部分を含めることができる．指定の方法は，oオプションのフィールド番号の並びに，辞書ファイルのコメントであれば‘rc’を，入力行のコメントであれば‘ic’を指定するだけである．

コメントを出力する場合のルールは以下の通り．まずoオプションの並びの中のより末尾に近い順で，それぞれ1回のみ，かつ他のフィールドの後方に出力される．

1. oオプションを指定した場合には，様々なフィールドが一行に並べられるので，辞書ファイルや入力ファイルのフィールド区切り記号は無視され，デフォルトで半角のカンマと空白の並び", "が区切り記号に使われる．また，出力の区切り記号列をOオプションによって指定することもできる．例えば，-O "; "や-O "\t"などと指定することが可能．
2. コメント以外の出力フィールドは，Oオプションで指定された区切り記号（指定しない場合は", "）で区切らるが，普通のフィールドの末尾とコメントの間の区切りは単なる半角の空白である．ただし，辞書ファイルのコメントと入力行のコメントがともに出力指定されている場合には，これら二つのコメントは上記Oオプションで指定された区切り記号で区切られる．

# 7　各種オプションについて

## 7.1　オプション形式

コマンドラインで与えることができる各種のオプションについて説明します．まずオプションの指定方法は以下のとおりです．長いオプション名でも短いオプション名でもかまいません．

```
ruby refsort.rb [--charcode]     [-a]
                [--evaluate]     [-e]
                [--rfile]        [-f]  filename
                [--rsep]         [-R]  "..."
                [--isep]         [-I]  "..."
                [--osep]         [-O]  "..."
                [--rfield]       [-r]  "?,?,?-?,..."
                [--ifield]       [-i]  "?,?,?-?,..."
                [--ofield]       [-o]  "??,??,??-?,..."
                [--numbered]     [-n]
                [--unique]       [-u]
                [--ignorehyphen] [-b]
                [--ignorecase]   [-c]
                [--squeezespace] [-s]
                [--detectdouble] [-d]
                [--milestone]    [-m]
                [--showencoding] [-g]
                [--quiet]        [-q]
                [--help]         [-h]
                [--version]      [-v]
                filename
```

## 7.2　各オプションの詳細

a 辞書ファイルを指定する代わりに，入力行をその実質名の文字コードに従って昇順に並べ替えて出力する．通常の sort コマンドと等価な機能である．

e 辞書ファイルを指定する代わりに，入力行の実質名を数値であると見なして，その大小に従って昇順に並べ替えて出力する．数値と見なされるのは，半角で書かれた整数や浮動小数点数で，指数表現も含めて通常の数値表現は全て変換可能である．全角の数字は数値と見なされない．数値に変換できない文字列はゼロと見なされる．

`f filename` 辞書ファイル`filename`を`f`の直後に指定する．`f`の指定が無かったり，`f`の後にファイル名が指定されておらず，`a`オプションも`e`オプションも指定されていないと直ちに処理を終了する．辞書ファイルはパス名を含めて指定することができる．

`R "..."` 辞書ファイルに有効なフィールド区切り記号を，オプション文字`R`の直後に二重引用符で囲んで列挙して定義する．区切り記号は全角半角を混在して複数書くことができる．普通文字は単にその文字を書くだけでよい．特殊文字としてタブを指定する場合には記号`\t`を用いる．デフォルトでは半角の空白とタブが指定されている．`R`オプションを指定すると，これらデフォルトの区切り記号は取り消されるので，全ての必要な区切り記号を改めて指定しなければならない．その他のフィールド分割の規則については第 3 章第 3.2 節の“**フィールド分割規則**”を参照．

この区切り記号の指定は，辞書ファイル中に置かれた埋め込みオプション`#!R`の指定によって変更することができる．また，`R`オプションや，埋め込みオプション`#!R`でデフォルト状態から変更した区切り記号を，デフォルト状態に戻したいときには，埋め込みオプションで空の文字列`""`を指定すればよい．

`I "..."` 入力ファイルに有効なフィールド区切り記号を，オプション文字`"I"`の直後に二重引用符で囲んで列挙して定義する．区切り記号は全角半角を混在して複数書くことができる．普通文字は単にその文字を書くだけでよい．特殊文字としてタブを指定する場合には記号`\t`を用いる．デフォルトでは半角の空白とタブが指定されている．`I`オプションを指定すると，これらデフォルトの区切り記号は取り消されるので，全ての必要な区切り記号を改めて指定しなければならない．その他のフィールド分割の規則については前記第 3.2 節の“**フィールド分割規則**”を参照．

この区切り記号の指定は，辞書ファイル中に置かれた埋め込みオプション`#!I`の指定によって変更することができる．また，`I`オプションや，埋め込みオプション`#!I`でデフォルト状態から変更した区切り記号を，デフォルト状態に戻したいときには，埋め込みオプションで空の文字列`""`を指定すればよい．

`O "..."` 出力に有効なフィールド区切り記号列をオプション文字`"O"`の直後に二重引用符で囲んで定義する．特殊文字としてタブを指定する場合には記号`\t`を用いる．デフォルトでは`",`
`"`という文字列が指定されている．

`r ...` 辞書ファイル中で実質名としてどのフィールドを使うか，その順序と番号を指定する．フィールド番号は 0 から始まる．辞書ファイル中の連続した区切り記号は空のフィールドを作る．デフォルトでは`-r 0`が指定されていると見なされる．

`i ...` 入力中で並べ替えの対象となる実質名としてどのフィールドを使うか，その順序と番号を指定する．デフォルトでは`-i 0`が指定されていると見なされる．

`o ...` 辞書ファイルと入力の中のどのフィールドをどのような順序で出力するのかをカンマで区切って（空白をはさまずに）指定する．フィールドの指定は，接頭辞‘`r`’と‘`i`’で辞書ファ

イルと入力を区別し，それに接する数字でフィールド番号を指定する．連続したフィールドはハイフンでその両端を示して指定することができる．またフィールド全体を指定する場合には‘ra’や‘ia’，コメントを指定する場合には‘rc’や‘ic’という記号を使うことができる．このオプションが指定されない場合には，出力は単に入力行そのものとなる．

`n` 出力に通し番号をつける．入力中に重複した実質名があったときには，重複した分はまとめて一つと数える．番号は半角 6 桁で出力される．

`u` 入力テキスト中から実質名だけが取り出され出力される．しかも重複した実質名があった場合にはそれらは一つの実質名として一行で出力される．`-i` オプションで複数のフィールドが指定されている場合には，各フィールドを出力区切り記号（デフォルトは`", "`）で区切って出力する．別名どうしが入力された場合には，それらは一つのレコードとして並べ替えられ，番号も一つしか与えられないが，出力時には入力された通りの見かけ上異なる文字列として表示される．

`b` 辞書ファイルと入力行の中で，半角のハイフン記号と半角の空白文字を同一視する．この処理はフィールド分割した後の実質名で行われる．

`c` 辞書ファイルと入力行の中で，半角の英文字について大文字と小文字の区別をしない．

`s` 辞書ファイルと入力行の中で，連続した空白を一つにまとめる．

`d` 入力テキスト中に重複した実質名があった場合に，その行番号と内容を標準エラー出力（コンソール画面）に表示する．

`m` 辞書ファイル中に埋め込みマイルストーンがあった場合には，規則に従い出力する．

`g` 辞書ファイルと入力それぞれのエンコーディングをコンソールに出力する．

`q` 警告文のうち，空フィールドの存在の警告文の標準エラー出力への出力を抑制する．

`h` Refsort のコマンドオプションを表示し，直ちに終了する．

`v` Refsort のバージョンを表示し，直ちに終了する．

- オプションを何も指定しないか，オプション指定に誤りがある場合には，Refsort は`usage`というメッセージを表示して直ちに終了する．
- 指定されたファイルが開けないなど，実行時に何らかの不都合があった場合，Refsort は適当なエラーメッセージを表示して直ちに終了する．

# 8 入出力指定

入力ファイルはコマンドラインの最後に指定します．例えば，

```
ruby refsort.rb -f jpblist_v70p2w.ref input.txt
```

ただし Refsort はフィルタとしても動作します．すなわち標準入力（通常はキーボード）からデータを読みとり，標準出力（通常はコンソール画面）に結果を出力します．従って入力ファイルを指定せずに

```
ruby refsort.rb -f jpblist_v70p2w.ref
```

と打ち込むと，Refsort は標準入力（キーボード）からの入力を待つので，Windows の場合には，

```
ハ シ ボ ソ ガ ラ ス<CR>   （<CR> は リ タ ー ン ・ キ ー）
ジ ョ ウ ビ タ キ<CR>
       :
       :
シ ジ ュ ウ カ ラ<CR>
c-z<CR>                （c-z は control-z）
```

と打ち込みます．最後の c-z はキーボード入力の終わりをシステムに伝えます．UNIX の場合には c-d に替えてください．Refsort は打ち込まれたリストを jpblist_v70p2w.ref の順番に従って並べ替え，画面に表示します．もちろんリダイレクト機能を使って標準入力をキーボードからファイルに切り替えてやることも可能です．これは記号 '<' を使って以下のように指定します．

```
ruby refsort.rb -f jpblist_v70p2w.ref < input.txt
```

こうすると，入力ファイルの内容があたかもキーボードから入力されたかのように処理されます．並べ替えられた結果は標準出力（コンソール画面）に表示されます．

表示された画面はいずれスクロールして消えて無くなり記録に取る事が出来ないので，画面の代わりにファイルに出力させるようにするのが標準出力のリダイレクトです．これは次のように記号 '>' を使って実現されます．

```
ruby refsort.rb -f jpblist_v70p2w.ref input.txt > output.txt
```

辞書ファイルや入力ファイルにエラーがなければ，今度は画面には最小限の情報しか表示されません．コマンド実行後にディレクトリを取ってみると指定した名前のファイルが作られているのがわかるはずです．TYPE コマンドなどで内容を確認できます．これは単なるテキストファイルなので，エディタなどで更に手を加える事も自由です．

更に高度な使い方にパイプがあります．パイプとはフィルタを複数個使って，入力を様々なフィルタの中を通して最終的に加工された結果を得るための機構で，これもリダイレクトと同じくコマンドコンソールの基本機能の一つです．パイプには記号 '|' を用います．具体的な使用例は後述します．

**■重要な注意**　Refsort/Ruby は Ruby 1.9 系列以降のエンコーディングの取り扱い規則に従いますので，入力された文字列のエンコーディングの区別は重要です．しかし，標準入力からの入力では，最初の行にエンコーディングを指定する `#!E` コメント行を入力することは**許可されません！**標準入力からの入力は，そのシステムでの最も自然な言語環境 Locale に従うのが当然であり，しかも Locale で定められているもの以外のエンコーディングの文字列をキーボードから入力することは簡単ではありません．従って，標準入力からの入力ではエンコーディングの明示的指定はできないと規定し，そのシステムで最も自然な言語環境 Locale に従ってエンコーディングが設定されます．このルールは，標準入力とみなされるリダイレクトされた入力やパイプでつながれた入力にも適用されますので，特にご注意ください．

# 9　各種辞書ファイルの紹介

## 9.1　日本産鳥類辞書 `jpblist_v70p2.ref`

バーダーの皆様の便宜を考えて，日本産鳥類のリスト `jpblist_v70p2.ref` を辞書ファイルとして別途提供しています．これは日本鳥学会の**日本鳥類目録 改訂第 7 版** *(2012)* をそのまま辞書ファイルとして編集しなおしたものです．この版では，科や目の変更をともなう大幅な改訂が行われ，また多数の亜種も収録されました．新しい分類に慣れない方も多い[*18]と思いますが，そのためにこそ Refsort/Ruby とこの辞書ファイルをお使いください．

この辞書のフィールドは，学名，英名，和名の順で 3 つ．フィールドの区切り記号は半角のカンマで，埋め込みオプションを用いて辞書ファイルの中で指定してありますので，利用者は辞書ファイルの区切り記号を意識する必要はありません．辞書ファイルは普通のテキストファイルなので可読です．すなわち，辞書ファイルの先頭部分からこれらの条件は簡単に読み取ることができます．区切り記号とフィールドの割り当ては特に重要なので，辞書ファイルを使うときにはまず先頭部分を読む習慣をつけると良いと思います．

大変残念なことに，和名，学名は特定できても，英名が定義されていない亜種が多数あります．これはそもそも英名が無いので仕方が無いのですが，この場合は英名は空欄にしてあります．また，英名は IOC World Bird Names の，2012 年秋の時点[*19]で最新だった v3.2 を採用しています

[*18] 実際，この新目録の出版から約 4 年以上が経っていますが，いまだに旧目録に則ったフィールド・ログを見かけます．長年使われてきた分類体系を変更することの難しさがよくわかります．一方で，図鑑やフィールドガイドの類はさすがに新しい分類体系に従った改訂版が出版され始めています．日本鳥類目録改訂第 7 版は，分類体系の変更だけではなく，多数の亜種が正式に収録されたことも特徴なので，これら図鑑類にもそれらが収録されるようになったことが従来との大きな変化点です．

[*19] 日本鳥類目録改訂第 7 版が出版された時点

ので，オリジナルの日本鳥類目録に収録されているものとは異なっている場合があります[*20]．

さらに細かい注意ですが，jpblist_v70p2.ref と総称される辞書ファイルの中で，エンコーディングが Windows-31J であるもののファイル名に‘w’という接尾辞を付けて jpblist_v70p2w.ref としています．同様にエンコーディングが UTF-8 であるものを jpblist_v70p2u.ref というファイル名にしてあります．これらの区別と使い分けにご注意ください．

また jpblist_v70p2.ref は Refsort 専用というわけではありません．単純なテキストファイルですので，簡便な鳥類の和名・英名・学名事典としても使用できます．Grep[*21]などの検索用コマンドを使ってお気軽にお使いください．

## 9.2 日本種子植物リスト jplant054.ref

旧 NIFTY-Serve FBIRD フォーラムで，1994 年に PALITAN さんが日本の種子植物のリスト PLANT03.REF を Refsort 用に公開されました．これは約 5,400 種の和名と学名が新エングラー体系で記載された大変な労作で，個人の手になるものとしては最大級のリストです．

学術の世界では，すでに新エングラー体系に代わる APG III という最新の遺伝子系統学に基づく分類体系が使われるようになっていますが，過去の膨大な記録を参照したり整合性を取るために，新エングラー体系も当分の間は（特にアマチュアの間では）使われることが予想されています．また，APG III に基づく辞書ファイルを作るのはあまりにも作業が膨大で，きちんとしたプロジェクトを組んでチームで行う必要があると思います．

このリストを Refsort/Ruby で扱いやすくするために編集しなおしたものが jplant シリーズで，埋め込みマイルストーン，埋め込みオプションや別名の機能，エンコーディング明示機能を使えるようにした最新のバージョンが jplant054.ref です．もちろん日本鳥類辞書の場合と同様に，区切り記号は埋め込みオプション #!R で指定してありますので，コマンドラインで改めて指定する必要はなく，辞書ファイルの区切り記号が何であるかを意識する必要はありません．また日本産鳥類辞書と同様，エンコーディングを Windows-31J としたファイルには‘w’という接尾辞を，UTF-8 としたファイルには‘u’という接尾辞をつけて区別しています．

この辞書ファイルは種数が膨大なだけに，細かなデバッグが不足しているのが欠点です．したがってベータ版の扱いとし，v0.54 としています．また植物の和名は別名が多く，また栽培種や外

[*20] さらに付け加えると，IOC List 自身が英名の改訂を続けていますので，最新版の IOC List とは異なっている可能性があります．ちなみに，2017 年 5 月時点での IOC List の最新版は v7.2 です．ずいぶん進みましたね．

[*21] Unix では非常によく知られた文字列検索用のユーティリティ・プログラムです．検索に正規表現が使えるので非常に強力ですが，コマンドラインでの使用が原則なので，初心者には取り付きにくいのが難点です．

来種，高山に自生する変種などの和名は混乱している面があり，この辞書のメンテナンスは悩みの種です．しかし，このように多様性の高い体系でのソートこそ Refsort が本来目指すべき領域なので，今後とも利用者の皆様のご協力を得ながら，改善していきたいと思います．

## 9.3 Sibley-Ahlquist 分類世界鳥類リスト `wbird_sa091a.ref`

このリストは，世界の鳥類約 1 万種を Sibley-Ahlquist 分類体系に従ってリストしたものです．作成に当たっては以下のような方針で臨みました．

まず jWiki[1]（Wikipedia 日本語版）の記事 “シブリー・アールキスト鳥類分類” の分類と，学名，英名，和名に従ってリストを作成しました．これが原型です．

次に**世界鳥類名勉強会**が作成した**世界の鳥リスト**（WBL[3] と略称）に記載されている学名，英名，和名との照合を行いました．この WBL の学名と英名は Birds of the World Ver.2 (Charles G. Sibley, 1996) に準拠しているとのことです．和名に関しては，世界鳥類和名辞典（山階芳麿, 1986），世界鳥類名検索辞典（白井祥平, 1992），日本鳥学会誌，野鳥誌，月刊誌 BIRDER 等の文献を参考にして作成されているそうですが，このグループが新たに創作した和名も含まれています．その結果，WBL のほうが jWiki よりも多くの和名を（それらの権威付けは別として）含んでいますので，jWiki のリストに WBL から和名を補足していって作ったのがこの辞書ファイルと考えて差し支えありません．

照合の結果，食い違いが発見された場合には，それぞれの種に関する jWiki と eWiki[2]（Wikipedia 英語版）の記事を相互に参照して，修正や補完を行いました．学名や英名のバリエーションはできる限り取り込みましたが，特に eWiki の記事で最新の分類学の見解が述べられている場合には，その見解に従って二つの種を一つに統合したり，亜種を種に格上げ，すなわち種を分割したり，逆に種を亜種に格下げしたり，という作業を行っています．この過程では，Avibase[11] (The World Bird Database) も頻繁に参照して和名の検索を行いました．少数ながら，BirdLife International (BLI[12]) の記事を参考にして修正を行った種もあります．また，明らかな誤植や誤記もこの時点で修正しました．

さらに，それでも和名が確定できない場合には，私自身が和名を創作しました．しかし，これらの作業は鳥類分類に関しては素人が独断を交えて行っています．さらに，全体で 1 万種ものリストをくまなく確認することは個人では不可能であり，上記の照合作業の中でも誤りを修正しきれていない恐れがあります．従って，このリストは v0.9 という扱いにしたいと思います．誤りの指摘や修正意見をお寄せいただければ幸いです．

Sibley-Ahlquist 分類は遺伝子解析を鳥類分類学を持ち込んだ画期的なものですが，内容はそれほ

ど支持されているわけではありません[*22]ので，このリストもどれほど役に立つのかわかりません．私としては相当の労力をつぎ込んで作成したリストではありますが，現在では**この辞書ファイルの保守・改訂は行っていません．**今後は IOC (International Ornithological Congress) が発行している “Birds of the World: Recommended English Names” (http://www.worldbirdnames.org/) に基づいたリストを主流の辞書ファイルにしていく方針です．

## 9.4 IOC World Bird Names 世界鳥類リスト ioclist_v72.ref

このリストは，IOC (International Ornithological Congress) が発行している “Birds of the World: Recommended English Names” (http://www.worldbirdnames.org/) の最新版を Refsort/Ruby の辞書として再編集したもので，世界の鳥類約 1 万 1 千種とそれらの亜種の学名と英名のリストです．分類体系としては伝統的なものに基づいているようですが，各所で最新の分類学の成果が取り入れられており，亜種から種に格上げされたものがかなりの数にのぼりますし，またある属に属する多くの種が異なる属に異動してしまうこともあります．

この IOC List は 3 ヶ月に一度という大変短い間隔で改訂が続けられており，学名や英名がその都度改訂されていきます．ついていくのが大変なほどですが，そのために特に前記の日本鳥類目録との食い違いはかなりの箇所にのぼります．これをどう解釈して使い分けていくかは難しい問題です．幸いなことに Refsort/Ruby には**別名**という柔軟性が用意してありますので，別名として両方の表記を併存させることが可能な場合が多くあります．食い違いに不便を感じる場合には，ご自分で辞書ファイルを編集して別名を加えることを提案したいと思います．

また，IOC List の編纂の主眼の一つは唯一性のある英名の提案なので，これまで複数の呼び方が並立していた英名を，バッサリと一つの名前でのみ記述しています．バリエーションの多い英名の使いにくさに業を煮やしてのことだと思います．例えばアビ属では英名の Diver を落とし，米名の Loon のみで記述されています．これが良いことなのか私にはよくわかりません．イギリスの鳥類学者やバーダーはどのように感じているのでしょうか？このような場合も別名を使うことで両方の名前を併記することは可能ですが，この辞書ファイルでは IOC List に忠実に，オリジナルのまま，すなわち Loon のみを収録しています．

IOC リストは IOC Master List と呼ばれるものが Excel のワークシート形式[*23]で提供されていますが，それを CSV ファイル[*24]に書き出し，さらに Ruby スクリプトを用いてほぼ自動的に

[*22] 例えば上記 Wikipedia 日本語版の記事をご覧ください．後のより精度が高い遺伝子解析結果と食い違う点があり，現在では Sibley-Ahlquist 系統は否定的に扱われています．しかし古典的な分類系統を進化系統学の考え方で変革しようとした姿勢は評価されているようです．

[*23] 当然のことながら 30,000 行を超える広大なワークシートで，検索機能を活用しないと目的の属や種を探し出すことは困難なほどです．ファイルサイズも xlsx 形式で 2MB 近くあります．

[*24] Comma Separated Value の意．フィールドをカンマで区切ってテキストファイルとして書き出す方法．カンマの代わりにタブで区切る TSV という書式も伝統的に使われています．

Refsort/Ruby の辞書ファイルを作ることができます．従って，IOC リストの更新に合わせてタイムリーに Refsort/Ruby の辞書を提供することができます[*25]．

注意点として，このリストのエンコーディングは UTF-8 を正とし，改行コードは LF，ファイル名には接尾辞 ‘u’ を付けています．これは，リスト中にウムラウトやアクセントを含むアルファベットが多用されているためです．英名であるにも関わらずドイツの人名がそのまま含まれている種，例えば "Seicercus soror, Alström's Warbler, ヒラオモリムシクイ" などというものがあるのです．ドイツ系アメリカ人の名前では o ウムラウトは “oe” に置き換えられていることが多く，アクセントを持たない英語のアルファベットで統一するというポリシーを採用しても良かったように思いますが，IOC リストは人名に関してはオリジナルの表記を尊重しているようで o ウムラウトがそのまま使われています．ということは，フランス語，スペイン語，北欧や東欧の言語に見られるアクセントや多様なアルファベットも人名に含まれて取り入れられるということになります．このため，そのような多様性に対応できるようエンコーディングは UTF-8 を正とすることにしました．

ただしユーザの便宜も考え，エンコーディングを純粋な US-ASCII に変換し，改行コードを CR/LF，ファイル名の接尾辞を ‘a’ としたものも提供しています．さらに，このファイルには**冒頭のエンコーディング指定を敢えて付けていません**．これにより，非常に幅広いプラットフォームでそれぞれのデフォルトの Locale での使用が可能となります．ただし，このファイルではオリジナル版に含まれているウムラウトやアクセントがそれらに最も近い文字に簡略化されており，厳密な意味では正しくありませんので，使用する際には注意してください．

## 9.5 IOC World Bird Names 世界鳥類リスト日本語版 ioclist_v72j.ref

前記 9.4 節の IOC World Bird Names に記載されている全ての種に和名を付けたものがこの辞書ファイルです．大半の和名は前記 9.3 節の Sibley-Ahlquist 分類世界鳥類リスト wbird_sa091a.ref から引っ張ってきたものですが，一部 jWiki や Avibase から引用したものもあります．それでも多数の未命名の種が出てきましたので，緊急避難として私自身が和名を創作しました．しかしこれはあくまで暫定的な処置と考えており，今後，ユーザーの方々からの修正意見を期待しています．和名の由来はできる限りコメント中 ‘##’ という記号の後に記載するようにしてありますので，参考にしてください．

また，第 4 章 “**辞書形式**” の第 4.3 節の最後で “**アマサギの例**”（17 ページ）の脚注にも書きましたが，IOC List ではかなり頻繁に種の分割や合併が行われ，それに伴って英名そのものの改訂が行われています．この場合に，和名を再定義するのかしないのか，するとすればどのような方針に

[*25] IOC Master List があれば辞書ファイルを完成させるのに 10 分とかかりません．

従って改定するのか，非常に悩ましい問題が発生します．これは私の手には負えませんし，そのような権威もありませんので，次回改訂の際の日本鳥学会での議論に期待したいと思います．

種とは異なり亜種の和名のほとんどは空欄のままですが，前記の日本鳥類目録改訂第 7 版に収録されている亜種で，この IOC List にも収録されている亜種は，和名に**亜種**という接頭辞をつけて収録しました．

和名追加のプロセスは各種文献や Web サイトの参照を含む大変手間のかかる作業であるため，残念ながらそれほどタイムリーにこの辞書ファイルをリリースすることはできません．できる限り自動化を行なってはいるものの，種の分割や属の異動があると完全な手作業になるので，オリジナル版のリリースに比べると，何週間か遅れてのリリースとなることをご容赦ください．

注意点として，このリストもエンコーディングは UTF-8 を正とし，改行コードは LF，ファイル名には接尾辞 ‘u’ を付けています．これは，リスト中にウムラウトやアクセントを含むアルファベットが多用されているためです．ただし利用者の便を考え，エンコーディングを Windows-31J に変更し，改行コードを CR/LF，ファイル名の接尾辞を ‘w’ としたものも提供しています．ただし，このファイルではオリジナル版に含まれているウムラウトやアクセントが最も近い文字に変更されており，厳密な意味では正しくありませんので，使用する際には注意してください．

# 10 使用例

## 10.1 最も基本的な例

まず，入力は以下のようなテキストファイルで test.txt と言う名前であると仮定します．

test.txt

```
# ----- test.txt -----
カモメ                    # 見誤り？
ツグミ                    # 鳴き声明瞭
オオタグロカモメ          # 見たことある？
セグロセキレイ
ヒヨドリ
マガモ
ヒバリ                    # 揚げヒバリ
タシギ
ミヤマセキレイ            # こんな鳥いたかしら？
ハシブトガラス            # 多数
ホオジロ
```

これに対して次のようなオプションを付けて実行します．

```
ruby refsort.rb -f jpblist_v70p2w.ref -n -r2 test.txt
```

辞書ファイル jpblist_v70p2w.ref のフィールド区切り記号は半角のカンマなのですが，これはすでに辞書ファイルの埋め込みオプション #!R によって指定されているので，改めて R オプションで指定する必要はありません．また，辞書ファイルの中で和名のフィールドは第 2 フィールドですから，フィールド番号 2 を指定する必要があります．さらに入力ファイルにはフィールドが第 0 フィールドのみ含まれていますが，そうすると I オプションや i オプションはデフォルトのままで良いはずですので，何も指定しません．すると次のような結果が得られます．

```
!R 52: -R "," redefined
jpblist_v70p2w.ref: 1145 records
!I 4: "オオタグロカモメ" not found in jpblist_v70p2w.ref
!I 10: "ミヤマセキレイ" not found in jpblist_v70p2w.ref
test.txt: 11 records
test.txt: 9 identified records
test.txt: 9 unique records
     1 マガモ
     2 タシギ
     3 カモメ                    # 見誤り？
     4 ハシブトガラス            # 多数
     5 ヒバリ                    # 揚げヒバリ
     6 ヒヨドリ
     7 ツグミ                    # 鳴き声明瞭
     8 セグロセキレイ
     9 ホオジロ
```

最初の 1 行は，辞書ファイル jpblist_v70p2w.ref の第 52 行目で，フィールド区切り記号が "," と再定義されていますよ，と注意を促すメッセージです．次の行は辞書ファイルを読み込んだ結果，1,145 個の実質名を認識した，というメッセージです．これは jpblist_v70p2w.ref の内容と整合が取れています．このメッセージは辞書ファイルを参照する場合には毎回必ず標準エラー出力（通常はコンソール画面）に出力されます．

次の 2 行は，入力ファイルの 4 行目と 10 行目に書かれている実質名が，辞書ファイル中に発見できなかったという警告のメッセージです．これでタイプミス[*26]がわかります．しかし，Refsort はこれら入力行を無視して処理を続行します．エラーメッセージや警告メッセージは標準エラー出力に出力されます．

さらに次の 3 行は，入力ファイルの中には合計で 11 個の実質名があり，その中で辞書ファイルの内容と照合できたものが 9 個，その中でさらに重複が無いものが 9 個あった，というメッセージです．このメッセージは毎回必ず標準エラー出力に出力されます．

[*26] **オオタグロカモメ**はおそらく**オオセグロカモメ**とタイプすべきところをまちがったのでしょう．**ミヤマセキレイ**はいかにもありそうな和名ですが，**ミヤマホオジロ**ならともかく，実際にはこのような種はありません．

次行以降は，並べ替えられた入力行で，標準出力（通常はコンソール画面）に出力されます．n オプションを付けたので，全部で 9 種類の鳥が記録されていることもわかります．この部分はファイルにリダイレクトすることができます．

## 10.2 複数ファイルの一括ソート

ある同一のフィールドに続けて 3 回行ったとします．そのときのそれぞれの探鳥リストが以下のような三つのファイル jan1591.txt，jan2691.txt と feb0391.txt に書かれていたとします．リスト中，第 0 フィールドは種名，第 1 フィールドの数字はその種のおおよその数，第 2 フィールドは日付を表しています．

jan1591.txt

```
# ----- jan1591.txt -----
#   Jan-15-1991      晴   気温約 9 度
#
マガモ                20       01/15/91
カワアイサ             2       01/15/91
カルガモ              50       01/15/91
ハシビロガモ           2       01/15/91
ヒドリガモ           200+      01/15/91
オナガガモ           150+      01/15/91
```

jan2691.txt

```
# ----- jan2691.txt -----
#   Jan-26-1991      晴   気温約 12 度
#
マガモ                10       01/26/91
カルガモ              30       01/26/91
ハシビロガモ          10       01/26/91
ヒドリガモ           100+      01/26/91
オナガガモ           100+      01/26/91
キンクロハジロ        20       01/26/91
```

feb0391.txt

```
# ----- feb0391.txt -----
#   Feb-03-1991      晴ときどき曇り   約 10 度
#
マガモ                20       02/03/91
スズガモ              50+      02/03/91
カルガモ              50+      02/03/91
アオサギ              20       02/03/91
ヒドリガモ            50+      02/03/91
オナガガモ           150+      02/03/91
```

三つのファイルをまとめて分類してみましょう．まず，コマンドコンソールのCOPYコマンドを使って，三つのファイルを連結させて一つのファイルにします．次に連結結果のファイル tmp.txt を Refsort で処理します．

```
copy jan1591.txt + jan2691.txt + feb0391.txt tmp.txt
ruby refsort.rb -f jpblist_v70p2w.ref -n -r2 tmp.txt
```

こうすると，重複した種については入力順，即ち今の場合には日付の順に並べられて以下のように出力されます．

```
!R 52: -R "," redefined
jpblist_v70p2w.ref: 1145 records
tmp.txt: 18 records
tmp.txt: 18 identified records
tmp.txt: 9 unique records
     1 ヒドリガモ          200+     01/15/91
       ヒドリガモ          100+     01/26/91
       ヒドリガモ           50+     02/03/91
     2 マガモ              20      01/15/91
       マガモ              10      01/26/91
       マガモ              20      02/03/91
     3 カルガモ            50      01/15/91
       カルガモ            30      01/26/91
       カルガモ            50+     02/03/91
     4 ハシビロガモ         2      01/15/91
       ハシビロガモ        10      01/26/91
     5 オナガガモ          150+     01/15/91
       オナガガモ          100+     01/26/91
       オナガガモ          150+     02/03/91
     6 キンクロハジロ       20      01/26/91
     7 スズガモ            50+     02/03/91
     8 カワアイサ           2      01/15/91
     9 アオサギ            20      02/03/91
```

## 10.3 多重キーによるソート

次に前出の tmp.txt をちょっと凝ったやり方で並べ替えてみましょう．まず，全体を出現数に関して昇順に並べ替えます．このときには目的のフィールドを数値の大小順に並べ替えるオプションである e を使いましょう．次にその結果を更に分類順に並べ替えて通し番号を振ります．そのためのコマンドは以下のようになります．

```
ruby refsort.rb -e -i1 tmp.txt \
    | ruby refsort.rb -f jpblist_v70p2w.ref -n -r2
```

ここでは Refsort を 2 回使い，パイプでつないでいます．画面には次のような結果が現れます．

```
1 ヒドリガモ          50+     02/03/91
  ヒドリガモ         100+     01/26/91
  ヒドリガモ         200+     01/15/91
2 マガモ              10      01/26/91
  マガモ              20      01/15/91
  マガモ              20      02/03/91
3 カルガモ            30      01/26/91
  カルガモ            50      01/15/91
  カルガモ            50+     02/03/91
4 ハシビロガモ         2      01/15/91
  ハシビロガモ        10      01/26/91
5 オナガガモ         100+     01/26/91
  オナガガモ         150+     01/15/91
  オナガガモ         150+     02/03/91
6 キンクロハジロ       20      01/26/91
7 スズガモ            50+     02/03/91
8 カワアイサ           2      01/15/91
9 アオサギ            20      02/03/91
```

種名は確かに分類順に並べられており，しかも重複して見られている種に関しては，数が少ない順に並べられていることがわかります．このように，それぞれ異なる基準を使って並べ替えを複数回行うことを多重キーによるソート[*27]と言います．一年分のフィールドノートの整理などに使えると思いますがいかがでしょうか？

このように，複数のフィールドをうまく使うことによって[*28]様々な分類が可能になります．皆さんも，どのようなフィールドノートを設計すれば，自分の希望通りの並べ替えができるか考えてみて下さい．

## 10.4　重複度の高いリストのソート

念願かなって秋の渡りの時期に伊良湖岬に行きました[*29]．午後のフィールドノートは以下のように書かれていました．ファイル名は `irago.txt` です．

---

[*27] 多重キーソートを行うには Refsort を 2 回以上続けて使う必要があります．ここではパイプを使ってつないでいますが，中間結果をファイルに書き出し，それに対して再度 Refsort を実行してもかまいません．

[*28] **複数のフィールドを指定しても多重キーソートにはならない**ことに注意してください．例えばコマンドオプションで，`-r 0,1 -i 0,1` と指定しても，第 0 フィールドと第 1 フィールドを並べた組みをソート対象にするだけで，フィールドの優先順位を指定できているわけではありません．

[*29] 私も徹夜で車を走らせて行ったことがあるのですが，この時は完全に空振りでした．タカ柱を見ようと張り切って行ったのですが，サシバはまばらにしか飛ばず，時々ヒヨドリの群れが海を渡る程度でがっかりしたことを覚えています．

irago.txt

```
# ----- irago.txt -----
# 伊良湖岬  10/10/1991
#
サシバ          1000+     13:00
ハチクマ          50+     13:15
サシバ           500      13:40
ハヤブサ          20      13:45
サシバ           500+     14:05
ハチクマ          10      14:10
サシバ           200+     14:20
ヒヨドリ         100+     14:50
サシバ           100      15:15
サシバ           300      15:50
```

これは時刻順に書かれたリストになっていますが，同じ種が何度も現れています．全体の種数は少ないのですが，重複度の高いリストであると言えます．これを種名ごとに並べ替えましょう．最も標準的なオプションでいいですね．

```
ruby refsort.rb -f jpblist_v70p2w.ref -nr2 irago.txt

irago.txt: 10 records
irago.txt: 10 identified records
irago.txt: 4 unique records
     1 ハチクマ          50+     13:15
       ハチクマ          10      14:10
     2 サシバ          1000+     13:00
       サシバ           500      13:40
       サシバ           500+     14:05
       サシバ           200+     14:20
       サシバ           100      15:15
       サシバ           300      15:50
     3 ハヤブサ          20      13:45
     4 ヒヨドリ         100+     14:50
```

なかなか見栄えのいいフィールドノートになったと思いませんか？サシバの数の時間的な変化がよくわかりますね．様々なカウント結果の整理にいかがでしょうか？

## 10.5　変則的な整理方法に利用する

フィールドノートの整理は何がなんでも分類順に種名を並べればいいというものではありません．特に何らかの目的を持って調査研究をしているときには，通常の分類順とは異なった角度からリストを眺めることも必要です．そういう時にも Refsort はお役に立てます．なぜかと言うと，辞書ファイルを自由に選択でき，必要とあれば辞書ファイル内の順番を変更することも自由だからで

す．その例を一つお目にかけましょう．

ある川の河口でフィールド調査をしたとします．河口と言っても広い地域ですから，ある鳥がどこで出たかも重要な情報になります．ある日のフィールドノートが次のようになっていたとします．

jan2291.txt

```
# ----- jan2291.txt -----
ミツユビカモメ          ( 20)   河口右岸
シロカモメ              (  5)   洋上
コアジサシ              ( 20)   河口両岸
コアジサシ              ( 10)   洋上
オオミズナギドリ        ( 50)   洋上
トビ                    (  5)   河口中州
ウミネコ                ( 50)   河口左岸
ウミネコ                ( 20)   洋上
セグロカモメ            (  2)   洋上
ミユビシギ              ( 20)   河口中州
ハシボソミズナギドリ    (  2)   洋上
オオセグロカモメ        ( 10)   河口左岸
シロチドリ              ( 10)   河口中州
カルガモ                ( 20)   河口左岸
カルガモ                ( 10)   河口中州
```

このリストでは第 3 列が環境を表しています．環境別に分類したリストを作ってみましょう．まず何種類の環境があるのかをリストアップしてみます．リストの中で，半角の括弧が意味を持つとフィールド番号がうまく定義できませんから，これらは区切り記号として追い出してしまいます．ここでは I オプションで 2 種類の区切り記号 (半角の括弧対) を指定しています．半角の空白を区切り記号に指定する必要はありません．なぜならば半角括弧に隣接する任意の個数の連続した半角空白は無視されるからです．

```
ruby refsort.rb -au -I "()" -i2 jan2291.txt > area.ref

jan2291.txt: 15 records
jan2291.txt: 15 identified records
jan2291.txt: 5 unique records
```

標準出力がリダイレクトされているので，画面には上の 3 行のメッセージしか出力されません．リダイレクト先の area.ref というファイルの中身は以下のようになります．u オプションを付けたので，実質名だけが取り出されて大変スリムなファイルになりました．

area.ref

```
# ----- area.ref -----
河口右岸
```

```
河口左岸
河口中州
河口両岸
洋上
```

全部で 5 種類の環境があることがわかりました．これを辞書ファイルとして使うことを考えます．但し，これでは単に文字コードの順序のままで不都合な点がありますから，エディタで行の順序を自分の目的に合わせて修正します．辞書ファイルの順序を自由に変更できる，これこそが Refsort/Ruby の真骨頂なのです．その例を以下に示します．

修正後の area.ref

```
# ----- area.ref -----
河口左岸
河口右岸
河口両岸
河口中州
洋上
```

これを辞書ファイルとして使うことにしましょう．そして，それぞれの環境ごとに出現した鳥を分類順に並べ替えて出力しましょう．再び多重キーによるソートの応用です．Refsort のコマンド行は以下のようにすればいいでしょう．

```
ruby refsort.rb -f jpblist_v70p2w.ref -r2 jan2291.txt \
    | ruby refsort.rb -I "()" -i2 -f area.ref
```

すると以下のようなリストが画面に表示されます．

```
!R 52: -R "," redefined
area.ref: 5 records
jpblist_v70p2w.ref: 1145 records
jan2291.txt: 15 records
jan2291.txt: 15 identified records
jan2291.txt: 12 unique records
stdin: 15 records
stdin: 15 identified records
stdin: 5 unique records
カルガモ                ( 20)   河口左岸
ウミネコ                ( 50)   河口左岸
オオセグロカモメ        ( 10)   河口左岸
ミツユビカモメ          ( 20)   河口右岸
コアジサシ              ( 20)   河口両岸
カルガモ                ( 10)   河口中州
シロチドリ              ( 10)   河口中州
ミユビシギ              ( 20)   河口中州
トビ                    (  5)   河口中州
オオミズナギドリ        ( 50)   洋上
```

```
ハシボソミズナギドリ      (  2)    洋上
ウミネコ                  ( 20)    洋上
シロカモメ                (  5)    洋上
セグロカモメ              (  2)    洋上
コアジサシ                ( 10)    洋上
```

いかがでしょうか？画面表示で jpblist_v70p2w.ref よりも area.ref が先に出てきているのはヘン？と思われるかもしれませんが，今は深く考えないことにしましょう．目的通りに並べ替えられているのがわかります．この後はエディタで環境別に切り分けるなり，考察を書き加えるなり作業を続けることが出来ます．色々なオプションを駆使すれば，あとはアイデア次第でかなり複雑な分類までもこなせます．

## 10.6 ログの中から鳥の名前を拾い上げてリストを作る

ある日の Web 掲示板で野鳥観察のフィールドノートが fieldnote.log というファイルに保存してあるとしましょう．この中から日本産鳥類の名前だけを拾いだし，それを分類順に並べてみましょう．かつての FBIRD のフィールドノートでは，鳥の名前は全角カタカナで書くのが決まりなので，全角カタカナだけからなる単語を拾ってそれを一行ごとに出力してやる必要があります．これには簡単なフィルタを Ruby で作ることにしましょう．スクリプトを kanapick.rb として添付してあります．ただし一つのカタカナ単語が一つの行内にあると仮定します．もしも単語が二行にまたがっているようだとうまくいきません．またエンコーディングは Windows-31J を仮定しています．

パイプを使ってつなぎましょう．2 行に分けて表示します．

```
ruby kanapick.rb fieldnote.log \
  | ruby refsort.rb -f jpblist_v70p2w.ref -nur2 > fieldnote.lis
```

このように，いろいろな人のフィールドノートが含まれているログを簡単に処理することができます．

## 10.7 さまざまな区切り記号に対応する

複数の人それぞれが，ある日の観察記録からフィールドノートを作りました．ところが事前に良く取り決めておかなかったために，フィールドの区切り記号がばらばらになってしまいました．各人のファイルをそのまま連結すると以下のようになります．

combined.txt

```
# ----- combined.txt -----
# 山田の記録（区切り記号は空白文字）
マガモ           10
カルガモ         30
ヒドリガモ       25
オナガガモ        8
#
# 田中の記録（区切り記号はセミコロン）
オカヨシガモ; 20
ホシハジロ; 10
ヒドリガモ; 20
カルガモ; 30
#
# 鈴木の記録（区切り記号はカンマ，2重引用符使用）
"ヒドリガモ",   20
"マガモ",       15
"オナガガモ",   20
```

しかしこのままではやりにくいので，入力のフィールド区切り記号をそれぞれ埋め込みオプションで指定しましょう．それは以下のようになります．

修正後の combined.txt

```
# ----- combined.txt -----
# 山田の記録（区切り記号は空白文字）
#!I ""
マガモ           10
カルガモ         30
ヒドリガモ       25
オナガガモ        8
#
# 田中の記録（区切り記号はセミコロン）
#!I ";"
オカヨシガモ; 20
ホシハジロ; 10
ヒドリガモ; 20
カルガモ; 30
#
# 鈴木の記録（区切り記号はカンマ，2重引用符使用）
#!I ","
"ヒドリガモ",   20
"マガモ",       15
"オナガガモ",   20
```

これを以下のコマンドでソートしてみます．

```
ruby refsort.rb -f jpblist_v70p2w.ref -nr2 combined.txt
```

すると出力は以下のようになります．

```
!R 52: -R "," redefined
jpblist_v70p2w.ref: 1145 records
!I 3: -I "" redefined
!I 10: -I ";" redefined
!I 17: -I "," redefined
combined.txt: 11 records
combined.txt: 11 identified records
combined.txt: 6 unique records
     1 オカヨシガモ; 20
     2 ヒドリガモ       25
       ヒドリガモ; 20
       "ヒドリガモ",    20
     3 マガモ           10
       "マガモ",        15
     4 カルガモ         30
       カルガモ; 30
     5 オナガガモ        8
       "オナガガモ",    20
     6 ホシハジロ; 10
```

分類は正しく行われていることがわかります．Refsort は o オプションや u オプションをつけない限り，入力行をそのまま書き出すことに注意してください．もしも u オプションをつけ加えると，第 0 フィールドのみが出力され，同一の種はまとめられるので，以下のような出力になります．

```
     1 オカヨシガモ
     2 ヒドリガモ
     3 マガモ
     4 カルガモ
     5 オナガガモ
     6 ホシハジロ
```

また，u オプションの代わりに -o i0-1 というオプションをつけて，入力行の第 0 フィールドと第 1 フィールドを出力するように指定すると，入力ファイルの区切り記号は一切現れず，o オプションに伴ってデフォルトで定義されている区切り記号 ", " で区切られった以下の結果が得られます．

```
     1 オカヨシガモ, 20
     2 ヒドリガモ, 25
       ヒドリガモ, 20
       ヒドリガモ, 20
     3 マガモ, 10
       マガモ, 15
     4 カルガモ, 30
       カルガモ, 30
     5 オナガガモ, 8
       オナガガモ, 20
     6 ホシハジロ, 10
```

今の例では入力行のフィールド区切り記号を変化させましたが，辞書ファイルに対しても全く同様のことができます．異なる区切り記号を使って作られた辞書ファイルを一つにまとめる場合には便利です．

また，埋め込みオプションにフィールド番号の再指定として r や i が使用できるようになりましたので，複数の辞書ファイルや入力ファイルをまとめる際の自由度がさらに高まりました．各人がばらばらの区切り記号やフィールドの順序でリストを作っていても，各作成者が責任を持って自分の区切り記号やフィールド番号を指定しておけばよいのです．

## 10.8 辞書ファイルの埋め込みマイルストーンを活用する

辞書ファイル jpblist_v70p2w.ref には埋め込みマイルストーンが仕掛けれられています．これを利用すると鳥類の分類の目や科を，ソート後の入力ファイルと同時に出力することができます．例えば，入力ファイルが以下のようになっていたとします．

showmilestone.txt

```
# ----- showmilestone.txt -----
ウミウ
カイツブリ
ハジロカイツブリ
キクイタダキ
カワウ
スズメ
オオセグロカモメ
アビ
マガモ
スズメ
カワラヒワ
マヒワ
```

これに対して以下のように m オプションを付けて実行します．

```
ruby refsort.rb -f jpblist_v70p2w.ref -r2 -mn showmilestone.txt
```

すると出力は以下のようになります．

```
!R 52: -R "," redefined
jpblist_v70p2w.ref: 1145 records
showmilestone.txt: 12 records
showmilestone.txt: 12 identified records
showmilestone.txt: 11 unique records
[ANSERIFORMES; カモ目]
[Anatidae; カモ科]
```

```
[Anas; マガモ属]    # "Linnaeus"
     1 マガモ
[PODICIPEDIFORMES; カイツブリ目]
[Podicipedidae; カイツブリ科]
[Tachybaptus; カイツブリ属] # "Reichenbach"
     2 カイツブリ
[Podiceps; カンムリカイツブリ属]    # "Latham"
     3 ハジロカイツブリ
[GAVIIFORMES; アビ目]
[Gaviidae; アビ科]
[Gavia; アビ属] # "Forster"
     4 アビ
[SULIFORMES; カツオドリ目]
[Phalacrocoracidae; ウ科]
[Phalacrocorax; ウ属]   # "Brisson"
     5 カワウ
     6 ウミウ
[CHARADRIIFORMES; チドリ目]
[Laridae; カモメ科]
[Larus; カモメ属]   # "Linnaeus"
     7 オオセグロカモメ
[PASSERIFORMES; スズメ目]
[Regulidae; キクイタダキ科]
[Regulus; キクイタダキ属]   # "Cuvier"
     8 キクイタダキ
[Passeridae; スズメ科]
[Passer; スズメ属]  # "Brisson"
     9 スズメ
       スズメ
[Fringillidae; アトリ科]
[Chloris; カワラヒワ属] # "Cuvier"
    10 カワラヒワ
[Carduelis; マヒワ属]   # "Brisson"
    11 マヒワ
```

## 10.9 表記の揺れへの対応

鳥の英名 (Common Name) には，地域によって様々なバリエーションがあり，しかも厄介なことに，数が不定の複数の単語から成っているうえに，単語間をハイフンでつなぐ場合とつながない場合とが共存しています．しかも，ハイフンの直後の文字を大文字にする場合と小文字にする場合の二通りの表記法があります．下にそのような例[*30]を示します．

[*30] このリストのうち，2 番めの Malayan Night-heron（ズグロミゾゴイ）という英名は，分類学者によっては Malaysian Night Heron と表記されていました，最近では IOC List のとおり Malayan Night Heron という表記が大勢のようです．さすがに，このようなスペリングそのものが異なる表記の揺れを Refsort のオプションで救済することはできません．辞書ファイルに別名として書き加えるべきです．

variation.txt

```
# ----- variation.txt -----
Pacific Reef Heron
Malayan Night-heron
Mountain Hawk-Eagle
Swinhoe's Storm-petrel
Sunda Scops-owl
```

これを以下のようなオプションでソートしてみましょう．英名は空白を含む複数の単語から成っているため，フィールド区切り記号がデフォルトの空白のままではまずいので，フィールド数は 1 なのですがおまじないとして I オプションで空白ではない区切り記号 "," を指定しています．また辞書ファイル jpblist_v70p2w.ref では英名は第 1 フィールドに書かれているので，r オプションでフィールド番号を指定します．さらに jpblist_v70p2w.ref では英名のフィールドには多数の空欄があって警告文が盛大に出力されるので，q オプションで警告文を抑制します．

```
ruby refsort.rb -f jpblist_v70p2w.ref \
    -nqr1 -I "," variation.txt
```

結果は以下のようになります．辞書ファイルと全く同じ表記でないものははねられてしまい，思ったとおりの結果にはなりません．また，辞書ファイルに 683 レコードしかないと表示され，これまで見てきた 1,145 レコードに満たないのて変に思うかもしれませんが，この辞書ファイルの第 1 フィールドには 683 個しか英名表記が無いためです．

```
!R 52: -R "," redefined
jpblist_v70p2w.ref: 683 records
!I 3: "Malayan Night-heron" not found in jpblist_v70p2w.ref
!I 5: "Swinhoe's Storm-petrel" not found in jpblist_v70p2w.ref
!I 6: "Sunda Scops-owl" not found in jpblist_v70p2w.ref
variation.txt: 5 records
variation.txt: 2 identified records
variation.txt: 2 unique records
     1 Pacific Reef Heron
     2 Mountain Hawk-Eagle
```

これに対して以下のようにハイフンを空白文字と同一視し，さらに大文字小文字を同一視する bc オプションを加えて実行してみましょう．

```
ruby refsort.rb -f jpblist_v70p2w.ref \
    -nqbc -r1 -I "," variation.txt
```

```
!R 52: -R "," redefined
jpblist_v70p2w.ref: 683 records
variation.txt: 5 records
```

```
variation.txt: 5 identified records
variation.txt: 5 unique records
     1 Swinhoe's Storm-petrel
     2 Malayan Night-heron
     3 Pacific Reef Heron
     4 Mountain Hawk-Eagle
     5 Sunda Scops-owl
```

めでたし，めでたしですね．辞書ファイルと入力ファイルの双方で，ハイフンをすべて空白に置き換え，大文字と小文字の区別をしないようにすると，たいていの場合はこのようにうまく行きます．

また，辞書ファイルと入力ファイルのいずれかで，単語間の空白の個数が食い違っている場合，例えば，"Pacific␣Reef␣Heron" と "Pacific␣Reef␣␣Heron" となっている場合は，非常に気づきにくいので何が間違っているのか分からず長時間悩むことがあります．このような場合には s オプションを使うと連続した複数の空白文字が一つにまとめられますので，タイプミスなどを救済することが出来ます．

## 10.10 植物観察記録への応用例

鳥の場合と異なり，植物の場合には何と言っても種類の数が膨大です．これを新エングラー体系[*31]と呼ばれる順番に並べ替えるのは結構な作業ですが，こういうときのためにこそ Refsort があるのです．例を見ていきましょう．

planttest0.txt

```
# ----- planttest0.txt -----
オオオナモミ
リュウノウギク
ノコンギク          # 1
アキノキリンソウ
オオアレチノギク
ヒメムカシヨモギ
コセンダングサ
アメリカセンダングサ
ヤクシソウ
セイタカアキノキリンソウ    # i.e. セイタカアワダチソウ
ノゲシ
セイヨウタンポポ
ダンドボロギク
ノコンギク          # 2
```

[*31] 最近では，新エングラー体系よりもさらに最新の進化系等学の結果を取り入れた APG III という分類体系が使われるようになって来ましたが，過去の膨大な記録との整合性を図るために，新エングラー体系は当分の間使われ続けるようです．

```
セイタカアワダチソウ
コウヤボウキ
ノハラアザミ
アキノノゲシ
```

このどこにでもあるような秋の植物観察リストを次のようにソートしてみます．実はこのリストには仕掛けがあって“セイタカアキノキリンソウ”は“セイタカアワダチソウ”の別名なのです．これはどのような結果になるのでしょうか？日本の種子植物の辞書である jplant054w.ref を用いましょう．植物辞書はやや大きなファイルなので，読み込み処理に少し時間がかかりますが，最近の PC であればほとんど気にならないでしょう．

```
ruby refsort.rb -n -f jplant054w.ref planttest0.txt

!R 111: -R "," redefined
jplant054w.ref: 5381 records
planttest0.txt: 18 records
planttest0.txt: 18 identified records
planttest0.txt: 16 unique records
     1 アキノキリンソウ
     2 セイタカアキノキリンソウ    # i.e. セイタカアワダチソウ
       セイタカアワダチソウ
     3 アキノノゲシ
     4 ノハラアザミ
     5 オオオナモミ
     6 ヤクシソウ
     7 リュウノウギク
     8 コウヤボウキ
     9 ノコンギク          # 1
       ノコンギク          # 2
    10 アメリカセンダングサ
    11 コセンダングサ
    12 ダンドボロギク
    13 セイヨウタンポポ
    14 ノゲシ
    15 オオアレチノギク
    16 ヒメムカシヨモギ
```

辞書ファイルには 5,381 種が登録されており，18 個の入力のうちユニークなものが 16 個であることも表示されています．2 番目の“セイタカアキノキリンソウ”と“セイタカアワダチソウ”が同一視され，9 番目の“ノコンギク”はダブって入力されていたわけです．仮に u オプションも併用すると，第 0 フィールドのみが出力され，ノコンギクは一つにまとめられますが，セイタカアキノキリンソウとセイタカアワダチソウは互いに別名ではあっても見かけ上異なる文字列であるので，それらは u オプションを付けない場合と同じように出力されます．

```
ruby refsort.rb -nu -f jplant054w.ref planttest0.txt
```

```
 1 ア キ ノ キ リ ン ソ ウ
 2 セ イ タ カ ア キ ノ キ リ ン ソ ウ
   セ イ タ カ ア ワ ダ チ ソ ウ
 3 ア キ ノ ノ ゲ シ
 4 ノ ハ ラ ア ザ ミ
 5 オ オ オ ナ モ ミ
 6 ヤ ク シ ソ ウ
 7 リ ュ ウ ノ ウ ギ ク
 8 コ ウ ヤ ボ ウ キ
 9 ノ コ ン ギ ク
10 ア メ リ カ セ ン ダ ン グ サ
11 コ セ ン ダ ン グ サ
12 ダ ン ド ボ ロ ギ ク
13 セ イ ヨ ウ タ ン ポ ポ
14 ノ ゲ シ
15 オ オ ア レ チ ノ ギ ク
16 ヒ メ ム カ シ ヨ モ ギ
```

また，埋め込みマイルストーンオプション `m` を使用すれば，辞書ファイルにコメントとして埋め込まれている科名，属名が出力されます．植物のソートでは，このように埋め込みマイルストーンや別名機能がお役に立てるのではないでしょうか？やってみてください．

## 10.11 IOC World Bird Names 世界鳥類リスト日本語版の使用

“IOC World Bird Names 世界鳥類リスト日本語版 `ioclist_v72j.ref` ” の項で述べたとおり，この辞書は IOC が定期的に改定している世界の鳥類リストで，それに和名のフィールドを加えたものです．この辞書ファイルを使ってみましょう．香港[*32]にバードウォッチングに行ったとします．だいぶ沢山の鳥を見てきたのですが，下記のようなフィールドノートができました．

maipo.txt

```
# ----- maipo.txt -----
コ サ ギ
チ ュ ウ サ ギ
ダ イ サ ギ
カ ワ ウ
ヘ ラ サ ギ
コ ガ モ
カ サ サ ギ
ク ビ ワ ガ ラ ス
```

[*32] 香港というとグルメとショッピングの観光地という先入観があるかもしれませんが，実はバーダーにとっては有名な**マイポ（米埔，*Mai Po*）**という水鳥のフィールドがあるところです．WWF 香港によって管理・保全されており，東アジアの水鳥の保護にとっても重要な場所の一つです．近年では対岸の深圳（シンセン）の環境汚染の影響が著しく，その保全が非常に憂慮されています．訪れるには WWF 香港などを通して事前の許可を取るか，WWF 香港のガイド付きツアーに参加する必要があります．

```
オナガ
アオハウチワドリ
アオサギ
ゴイサギ
カルガモ
チョウゲンボウ
カラフトワシ
アカガシラサギ
シロハラクイナ
ギンムクドリ
マミハウチワドリ
キマユムシクイ
カイツブリ
トビ
ノスリ
シマキンパラ
ダイゼン
アオアシシギ
ソリハシセイタカシギ
クロツラヘラサギ
ハシビロガモ
ユリカモメ
カノコバト
オナガガモ
オニカッコウ
ヒメヤマセミ
ノビタキ
カワセミ
ヤマショウビン
     ... 以下省略
```

これを以下のようなコマンドラインで並べ替えてみましょう．

```
ruby refsort.rb -f ioclist_v72jw.ref -mnq -r2 maipo.txt
```

すると以下のような出力が得られます．ちょっと長いですが一通り眺めてみてください．ちなみに，この辞書ファイルには 11,256 種の和名が収録されていることもわかります．

```
!R 66: -R "," redefined
ioclist_v72jw.ref: 11256 records
maipo.txt: 63 records
maipo.txt: 63 identified records
maipo.txt: 63 unique records
[NEOGNATHAE]
[ANSERIFORMES]  # [カモ目]
[Anatidae]      # [Ducks, Geese and Swans; カモ科]
[Anas]  # [マガモ属] # "Linnaeus, 1758"
     1 カルガモ
     2 ハシビロガモ
     3 オナガガモ
```

```
     4 シマアジ
     5 コガモ
[NEOAVES]
[PODICIPEDIFORMES]      # [カイツブリ目]
[Podicipedidae] # [Grebes; カイツブリ科]
[Tachybaptus]   # [カイツブリ属] # "Reichenbach, 1853"
     6 カイツブリ
[PELECANIFORMES]        # [ペリカン目]
[Threskiornithidae]     # [Ibises, Spoonbills; トキ科]
[Platalea]      # [ヘラサギ属] # "Linnaeus, 1758"
     7 ヘラサギ
     8 クロツラヘラサギ
     ... 以下省略
```

## 10.12 出力フィールドの指定

古い版の Refsort では，出力には入力ファイルの行全体か，u オプションを指定した場合には指定された実質名だけしか出力することができませんでした．Refsort/Ruby v2.00 からはこの制限を取り除き，辞書ファイルや入力の任意のフィールドを任意の順序で並べて出力することができるようになりました．またそのときのフィールド区切り記号も自由に指定できます．これにより，和名だけが書かれているフィールドノートに学名や英名を付加したり，あるいは新たな辞書ファイルを簡単に作ることができるようになりました．この章の最初の例 10.1 と同じファイル test.txt を使いましょう．

test.txt

```
# ----- test.txt -----
カモメ                  # 見誤り？
ツグミ                  # 鳴き声明瞭
オオタグロカモメ        # 見たことある？
セグロセキレイ
ヒヨドリ
マガモ
ヒバリ                  # 揚げヒバリ
タシギ
ミヤマセキレイ          # こんな鳥いたかしら？
ハシブトガラス          # 多数
ホオジロ
```

これに対して，以下のようなオプションを指定してみましょう．

```
ruby refsort.rb -f jpblist_v70p2w.ref -r2 \
    -O ",\t" -o ia,r0-1 test.txt
```

出力のフィールド区切り記号としてカンマに続いてタブを指定し，出力すべきフィールドとして

は，まず入力の全フィールド，それから辞書ファイルの第 0 フィールドと第 1 フィールドを並べて出力するよう指定しています．このコマンドの出力は以下のようになります．

```
マガモ, Anas platyrhynchos,     Mallard
タシギ, Gallinago gallinago,    Common Snipe
カモメ, Larus canus,    Mew Gull
ハシブトガラス, Corvus macrorhynchos,   Large-billed Crow
ヒバリ, Alauda arvensis,        Eurasian Skylark
ヒヨドリ,       Hypsipetes amaurotis,   Brown-eared Bulbul
ツグミ, Turdus naumanni,        Naumann's Thrush
セグロセキレイ, Motacilla grandis,      Japanese Wagtail
ホオジロ,       Emberiza cioides,       Meadow Bunting
```

いかがでしょうか？入力には和名しか書いてないのですが，学名と英名がちゃんと加えられて，見栄えのするリストになりました．辞書ファイルで別名が指定されている場合には，それがそのまま別名の区切り記号 '|' で区切られて出力されることにもご注意ください．

さて，出力フィールドには，辞書ファイルや入力行のコメント部分を指定することもできます．辞書ファイルのコメント部分を出力に加えるには 'rc' を，入力行のコメント部分を出力に加えるには 'ic' を o オプションに記述してください．ただし，コメント部分は以下のような規則で出力されることにご注意ください．例も示します．

1. o オプションにどのような順序で記載されていても，必ず普通のフィールドの後ろにしか出力されず，しかも，辞書ファイルと入力行の両方のコメントが指定されていた場合は，o オプションの並びでより末尾に近いものそれぞれ 1 個のみが選ばれ，その順序で出力される．
2. 普通のフィールドは o オプションで指定された区切り記号（指定しない場合は ",␣"）で区切られて出力されるが，コメントは，最後の普通のフィールの後に半角の空白 1 文字を置いて # 記号から出力される．また，辞書ファイルと入力行の両方のコメントを出力するように指定した場合には，これら二つのコメントの間は o オプションで指定した区切り記号で区切られる．

```
ruby refsort.rb -f jpblist_v70p2w.ref -r2 -o ia,ic,r0 test.txt

マガモ, Anas platyrhynchos
タシギ, Gallinago gallinago
カモメ, Larus canus # 見誤り？
ハシブトガラス, Corvus macrorhynchos # 多数
ヒバリ, Alauda arvensis # 揚げヒバリ
ヒヨドリ, Hypsipetes amaurotis
ツグミ, Turdus naumanni # 鳴き声明瞭
セグロセキレイ, Motacilla grandis
ホオジロ, Emberiza cioides
```

# 11　最後に

20 数年前に初めて REFSORT.EXE を書いたときに比べると，今の PC の利用可能なリソースは恐ろしいほどの質と量を持っています．リソースの大部分は GUI などに使われているのですが，強大なリソースをもったおかげで，二昔前だったら実用的ではなかったスクリプト言語によるプログラミングが脚光を浴びるようになってきました．Ruby は最も有望な言語の一つであり，日本人の手によるものです．今回 Refsort/Ruby のプログラミングを通して，その強力さと使いやすさを楽しみました．

辞書ファイルさえ用意すればさまざまな応用が可能です．このドキュメントでは主として分類学的な応用例を示しましたが，汎用性を重視して作成したプログラムなので，これ以外にも色々な用途があることと思います．是非チャレンジしていただきたいと思います．

最後に，Ruby の作者**まつもとゆきひろ**さん，getoptlong.rb の作者である**笠原基之**さん，そして 20 数年前に REFSORT.EXE を使っていただいた FBIRD の皆様に感謝します．

2017 年 5 月俊（とし）

# 参考文献

[1] Wikipedia Japanese version (http://ja.wikipedia.org/)
[2] Wikipedia English version (http://en.wikipedia.org/)
[3] World Bird List **世界の鳥** 世界鳥類名勉強会 編
(明石のはらくらぶ — http://www.eonet.ne.jp/~saezuri/)
[4] Birds of the World, Ver.2 (Charles G. Sibley, 1996)
[5] 世界鳥類和名辞典 (山階芳麿，1986)
[6] 世界鳥類名検索辞典 (白井祥平, 1992)
[7] 日本鳥学会誌, 日本鳥学会
[8] 月刊 Birder, 文一総合出版
[9] 野鳥, 日本野鳥の会 会員誌
[10] メジロ類の見分け方, 全国密猟対策連絡会議 20xx
[11] The World Bird Database (http://avibase.bsc-eoc.org/)
[12] Bird Life International (http://www.birdlife.org/)

# 索引