

# pBibTeX / upBibTeX マニュアル

日本語 TeX 開発コミュニティ\*

version 0.99d-j0.36, 2023 年 2 月 25 日

pBibTeX と upBibTeX は、それぞれ pTeX と upTeX と組み合わせて使用することを想定して開発された BibTeX の日本語対応版である。

- pBibTeX, upBibTeX の開発元：  
<https://github.com/texjporg/tex-jp-build/>
- upBibTeX の開発元 (j0.35 以前)：  
<http://www.t-lab.opal.ne.jp/tex/uptex.html>
- 本ドキュメントの開発元：  
<https://github.com/texjporg/pbibtex-manual/>

BibTeX の日本語化は、電力中央研究所の松井正一氏によって jBibTeX という名称で公開されたもの<sup>\*1</sup>がベースであり、その仕様は [1] に詳しい。また、オリジナルの BibTeX 付属ドキュメント `btxdoc.pdf` と `btXHak.pdf` を日本語に訳し、jBibTeX について補足を加えたドキュメントも用意されている [2, 3]。これらのドキュメントは、レガシーエンコーディングの扱いに関する記述（第 4 節を参照）を除き現在の (u)pBibTeX でも有効であるので、参照されたい。

本文書では、オリジナルの BibTeX の仕様を把握している読者を想定し、pBibTeX および upBibTeX における機能の変更・追加点を説明する。

---

\* <https://texjp.org>, e-mail: [issue\(at\)texjp.org](mailto:issue(at)texjp.org)

<sup>\*1</sup> 最終版は jBibTeX 0.31 である。1991/01/01 付の jBibTeX 0.30 のパッケージに、1992/10/31 に `fj.comp.texhax` グループへ投稿されたバグ修正パッチ (0.31) を当てて得られる。

## 目次

|     |                          |   |
|-----|--------------------------|---|
| 1   | 日本語化の仕様                  | 3 |
| 1.1 | 多バイト文字の扱い . . . . .      | 3 |
| 1.2 | 文字種が増えたことへの対応 . . . . .  | 5 |
| 1.3 | 日本語文字とそれ以外の区別 . . . . .  | 6 |
| 1.4 | その他詳細 . . . . .          | 7 |
| 2   | コマンドラインオプション             | 7 |
| 3   | kpathsearch ライブラリ変数      | 7 |
| 4   | 参考：jBibTEX と pBibTEX の違い | 8 |

# 1 日本語化の仕様

$\text{BnT}\text{E}\text{X}$  の日本語化の特徴は、大きく分けて3つである。

- 多バイト文字の扱い
- 文字種が増えたことへの対応
- 日本語文字とそれ以外の区別

## 1.1 多バイト文字の扱い

(u)p $\text{BnT}\text{E}\text{X}$  は、 $\text{BnT}\text{E}\text{X}$  から最小の変更量で日本語を含む文献情報を扱えるようにした都合上、多バイト文字の扱いは以下のとおりとなっている。

- 文字列の位置や長さは「文字単位」ではなく「バイト単位」でカウントする。
- その結果として、開始位置や終了位置が多バイト文字の途中となる場合は、**多バイト文字の途中で切られないように位置を調整する。**

具体的には、以下のスタイルパラメータが該当する。

- `substring$`

整数値2つ（長さ、起点）と文字列リテラル1つを `pop` し、指定の長さ（バイト長）の部分文字列を `push` する組込関数である。起点に指定された整数値が正であれば文字列の先頭から、負であれば末尾から長さを数えて取り出す。

多バイト文字についての開始・終了位置の調整は以下の順で行う。

- (1) 開始位置が多バイト文字の2バイト目以降であれば、1バイト目から取り出す。
- (2) 終了位置が多バイト文字の最終バイトでなければ、最終バイトまで取り出す。

したがって、この時点で p $\text{BnT}\text{E}\text{X}$ （内部コード `euc`）では最大2バイト長く、up $\text{BnT}\text{E}\text{X}$ （内部コード `uptex`）では最大6バイト長い文字列が取り出されうる。

さらに、(u)p $\text{BnT}\text{E}\text{X}$  j0.34 ( $\text{T}\text{E}\text{X}$  Live 2022) 以降では

- (3) 起点が #2 以上に指定されたにもかかわらず、上の (1) の調整により開始位置が文字列の先頭になってしまった場合は、その先頭が多バイト文字を切り捨てる。
- (4) 起点が #-2 以下に指定されたにもかかわらず、上の (2) の調整により終了位置が文字列の末尾になってしまった場合は、その末尾が多バイト文字を切り捨てる。


という調整も加えてある。したがって、p $\text{BnT}\text{E}\text{X}$  では最大1バイト短く、up $\text{BnT}\text{E}\text{X}$  では最大3バイト短くなる（場合によっては空になる）ことがある。

- `text.prefix$`

整数値（長さ）と文字列リテラルを `pop` し、文字列の先頭から指定の長さ（バイト長）の連続した文字列を `push` する組込関数である。終了位置が多バイトの文字の途中にならないよう調整されるので、p $\text{BnT}\text{E}\text{X}$ （内部コード `euc`）では最大1バイト長く、up $\text{BnT}\text{E}\text{X}$ （内部コード `uptex`）では最大3バイト長い文字列が取り出されうる。

以下に例を示す. `pBibTeX` と `upBibTeX` では同じ文字でもバイト長が異なる場合があり, その結果として取り出される見かけの文字数が異なることに注意.

| 例1:長めに調整(2)                |                  | <code>pBibTeX</code> | <code>upBibTeX</code> |
|----------------------------|------------------|----------------------|-----------------------|
| "あいうえお" #1 #1 substring\$  | 最初の 1 文字         | 「あ」                  | 「あ」                   |
| "あいうえお" #1 #2 substring\$  | 最初の 2 文字         | 「あ」                  | 「あ」                   |
| "あいうえお" #1 #3 substring\$  | 最初の 3 文字         | 「あい」                 | 「あ」                   |
| "あいうえお" #1 #4 substring\$  | 最初の 4 文字         | 「あい」                 | 「あい」                  |
| "あいうえお" #1 #5 substring\$  | 最初の 5 文字         | 「あいう」                | 「あい」                  |
| "あいうえお" #1 #6 substring\$  | 最初の 6 文字         | 「あいう」                | 「あい」                  |
| 例2:長めに調整(1)                |                  | <code>pBibTeX</code> | <code>upBibTeX</code> |
| "あいうえお" #-1 #1 substring\$ | 最後の 1 文字         | 「お」                  | 「お」                   |
| "あいうえお" #-1 #2 substring\$ | 最後の 2 文字         | 「お」                  | 「お」                   |
| "あいうえお" #-1 #3 substring\$ | 最後の 3 文字         | 「えお」                 | 「お」                   |
| "あいうえお" #-1 #4 substring\$ | 最後の 4 文字         | 「えお」                 | 「えお」                  |
| "あいうえお" #-1 #5 substring\$ | 最後の 5 文字         | 「うえお」                | 「えお」                  |
| "あいうえお" #-1 #6 substring\$ | 最後の 6 文字         | 「うえお」                | 「えお」                  |
| 例3:先頭文字の途中調整(3)            |                  | <code>pBibTeX</code> | <code>upBibTeX</code> |
| "あいうえお" #1 #6 substring\$  | 最初の 6 文字         | 「あいう」                | 「あい」                  |
| "あいうえお" #2 #5 substring\$  | 最初の 1 文字を除く 5 文字 | 「いう」                 | 「い」                   |
| "あいうえお" #3 #4 substring\$  | 最初の 2 文字を除く 4 文字 | 「いう」                 | 「い」                   |
| "あいうえお" #4 #3 substring\$  | 最初の 3 文字を除く 3 文字 | 「いう」                 | 「い」                   |
| "あいうえお" #5 #2 substring\$  | 最初の 4 文字を除く 2 文字 | 「う」                  | 「い」                   |
| "あいうえお" #6 #1 substring\$  | 最初の 5 文字を除く 1 文字 | 「う」                  | 「い」                   |
| 例4:末尾文字の途中調整(4)            |                  | <code>pBibTeX</code> | <code>upBibTeX</code> |
| "あいうえお" #-1 #6 substring\$ | 最後の 6 文字         | 「うえお」                | 「えお」                  |
| "あいうえお" #-2 #5 substring\$ | 最後の 1 文字を除く 5 文字 | 「うえ」                 | 「え」                   |
| "あいうえお" #-3 #4 substring\$ | 最後の 2 文字を除く 4 文字 | 「うえ」                 | 「え」                   |
| "あいうえお" #-4 #3 substring\$ | 最後の 3 文字を除く 3 文字 | 「うえ」                 | 「え」                   |
| "あいうえお" #-5 #2 substring\$ | 最後の 4 文字を除く 2 文字 | 「う」                  | 「え」                   |
| "あいうえお" #-6 #1 substring\$ | 最後の 5 文字を除く 1 文字 | 「う」                  | 「え」                   |

 (u)`pBibTeX` j0.33 (`TeX Live 2021`) 以前の `substring$` では, オリジナルの `jBibTeX` と同様, 上記のうち調整 (1) と (2) しか行っていなかった. すなわち「開始位置や終了位置が多バイトの文字の途中となる場合は, 位置を調整して常に長めに切り出す」という仕様であった. しかし, これでは

```
" (あいうえお" #2 global.max$ substring$ % 最初の1文字を除去したい
"あいうえお" #-2 global.max$ substring$ % 最後の1文字を除去したい
```

のような場合に文字を除去できず, 例えば「while\$ ループ内で `substring$` 関数により文字列を 1 つずつ切り詰めて短くする処理」に和文文字リテラルが渡ると文字列が一向に短くならず, 無限

ループが起きてしまっていた [7]. そこで, (u)pB<sub>W</sub>TeX j0.34 (TeX Live 2022) 以降は調整 (3) と (4) を加え, カウント起点が「先頭が多バイト文字の途中」または「末尾が多バイト文字の途中」の場合に限って取り除くことで短めに切り出すこととした. なお, カウント終点側の多バイト文字については常に長めに切り出すし, カウント起点が先頭でも末尾でもない多バイト文字の場合は長めに切り出すという点は従来と同じである. 上記の例を見るのが早いであろう.

- `int.to.chr$, chr.to.int$`

`int.to.chr$` は整数値 1 つを `pop` し, 対応する文字コードの文字 1 字を `push` する組込関数である. `chr.to.int$` は文字 1 字を `pop` し, 対応する文字コードの整数値 1 つを `push` する組込関数である. (u)pB<sub>W</sub>TeX j0.34 (TeX Live 2022) 以前では `int.to.chr$` は ASCII の範囲の整数 (最大 127) のみ有効, `chr.to.int$` は多バイト文字の場合先頭バイトのコード値が返る仕様であった.

(u)pB<sub>W</sub>TeX j0.35 (TeX Live 2023) 以降ではこれらの組込関数を文字集合全体が扱えるように拡張した. すなわち, 文字 1 字分<sup>2</sup>の多バイト文字列を対象とし, 文字コードの整数としては pB<sub>W</sub>TeX では JIS コード<sup>3</sup>, upB<sub>W</sub>TeX では Unicode のスカラー値を用いる.

## 1.2 文字種が増えたことへの対応

- `add.period$`

文字列リテラルを `pop` し, 最後の文字 (} を除く) がピリオド類 `<.><?><!>` のいずれでもないときに `<.>` を最後に加える組込関数である.

pB<sub>W</sub>TeX では, 全角の `<!><?><.><。>` (それぞれ U+FF01, U+FF1F, U+3002, U+FF0E) もピリオド類とみなし, これらで終わっても `<.>` を付加しない.

upB<sub>W</sub>TeX ではさらに U+203C, U+203D<sup>4</sup>, U+2047, U+2048, U+2049 もピリオド類とみなす.

- `format.name$`

文字列 (フォーマット指定), 整数値 (何番めか), 文字列 (名前リスト) を `pop` し, フォーマットされた名前の文字列を `push` する組込関数である.

(u)pB<sub>W</sub>TeX では, 日本人の姓名の間のスペースとして全角空白 U+3000 も半角空白と同じとみなし (全角空白は半角空白に変換して処理), また複数の氏名間の区切りとして `and` と同様に全角の読点 `<、>` とコンマ `<、>` (それぞれ U+FF0C, U+3001) も使える.

- `change.case$`

変換対象の文字列と変換方法の文字を `pop` し, 大文字小文字変換を施した文字列を `push` する組込関数である. 変換方法の文字は `l, u, t` で, それぞれ小文字への変換, 大文字への変換, タイトルケース (語頭が大文字で他が小文字) への変換を示す. もしこれが不都合で文字列の一部を変換したくない場合は, 対象の部分を波括弧 `{ と }` で囲むことにより変換なしに出来る.

---

<sup>2</sup> upB<sub>W</sub>TeX ではより正確にはコードポイント 1 つ分となる. Unicode では合成文字や異体字セレクタ等では複数のコードポイントで 1 文字となるため.

<sup>3</sup> JIS コードを 16bit 表現した整数値 ((区 + 32) × 256 + (点 + 32)).

<sup>4</sup> U+203D をピリオド類とみなす仕様はバージョン u1.29 (TeX Live 2023) 以降.

従来は大文字小文字変換が行われる文字集合は ASCII のアルファベットに限られていたが、`upBibTeX` バージョン `u1.29` (`TeX Live 2023`) 以降ではさらに Latin-1, Latin Extended-A, Greek and Coptic, Cyrillic, Cyrillic Supplement ブロックの文字も変換対象とするように拡張<sup>\*5</sup>した。

### 1.3 日本語文字とそれ以外の区別

- `is.kanji.str$`

(u)pBibTeX 独自の組込関数である。スタックトップの文字列リテラルを `pop` し、文字列中に「日本語文字」が 1 つでも含まれていれば整数値 1 を、含まれなければ 0 を `push` する。なお、「日本語文字」かどうかの判定は以下のように行う。

- pBibTeX の場合

ASCII の範囲 (0–127) に収まらない文字を全て「日本語文字」として扱う。

- upBibTeX (内部コード `uptex`) の場合

漢字・かな・ハングルに該当する Unicode ブロックの文字を「日本語文字」として扱う<sup>\*6</sup>。pBibTeX とは異なり、記号類 (句読点, 括弧類, ●○■□◆◇など) は「日本語文字」として扱わない (バージョン `u1.27` (`TeX Live 2021`) 以降 [5])。

(u)pBibTeX 用のスタイルファイルでは、しばしば著者名のフォーマットで「日本人の姓名の間にはスペースを入れない」という挙動を実現するために `is.kanji.str$` 関数が実用されているし、他にも “, et al.” の代わりに「ほか」としたり “Vol. 2” の代わりに「第 2 巻」としたりといったローカライズにも多用されている。

⚠ `is.kanji.str$` について、pBibTeX は松井氏の jBibTeX の仕様を踏襲している。upBibTeX のバージョン `u1.27` 以降では上記の調整を行っているが、これは

- `is.kanji.str$` が「日本語文字」が最低 1 つでも含まれていれば真となること
- 各種スタイルファイルで「著者名の文字列について `is.kanji.str$` が真ならば姓名の間にスペースを入れないという使われ方が多いこと

を踏まえたものである。少なくともラテン文字 (15) は ASCII の範囲 (0–127) に収まらないが欧文扱いされたほうが都合がよいし、さもないと Erwin Schrödinger のような姓名の間のスペースが失われてしまう。また、記号類 (18) の一部は欧文の文脈で使われる可能性もある<sup>\*7</sup>。上記の調整を行うことで、文字列全体で見たときに「日本語の文脈なら真、そうでなければ偽」となるので都合がよいという算段である。なお、漢字・かな・ハングルの文字で真としているのは、日本人・中国人・韓国人の姓名の間にはスペースを入れない記法がよく行われることに依る。

<sup>\*5</sup> pBibTeX ではギリシャ文字・キリル文字を全角文字として扱ってきた長年の慣習に従い、ギリシャ文字・キリル文字の大文字小文字変換は行わない仕様を維持している。また pBibTeX, upBibTeX ともに全角ラテン文字 (U+FF21..FF3A, U+FF41..FF5A, A B C など) の大文字小文字変換は行わない。

<sup>\*6</sup> 実装上は upTeX の `\kcatcode` と同じブロック分けを流用しているのでそれに即して記述すると、既定値が 16 (kanji), 17 (kana), 19 (hangul) のブロックを真, 15 (latin), 18 (CJK symbol) のブロックを偽としている。

<sup>\*7</sup> さらに言えばギリシャ文字とキリル文字も一部 JIS X 0208 に含まれ、(u)pTeX の既定で和文扱いされる (いわゆる全角文字である) し、upTeX での `\kcatcode` はともに 18 で記号類と同じ扱いである (ちなみに pTeX ではギリシャ文字は 17, キリル文字は 18)。

## 1.4 その他詳細

(u)pBibTeX は、文献データベース (BIB ファイル) を読み込んで文献リスト (BBL ファイル) を出力するが、ファイルの行末の扱いは以下のとおりである。

- 入力 **BIB** ファイル内の改行は、(u)pBibTeX では一律に半角空白と同様に扱われる。すなわち、(u)pTeX とは異なり日本語文字直後の改行も半角空白と等価である。
- **BBL** ファイルの出力時に、長いエントリを一行に出力するか複数行に分割するか決定する場面において、オリジナルの BibTeX は任意の半角空白を分割可能箇所とみなす<sup>\*8</sup>が、(u)pBibTeX j0.34 (TeX Live 2022) 以降は日本語文字の直後でない半角空白のみを分割可能箇所とみなす。これは、元々 BIB ファイルにあった日本語文字直後の半角空白が仮に BBL ファイルで改行に置換されれば、(u)pTeX での読込時に「日本語文字直後の改行は空白を発生しない」という仕様により半角空白が消えてしまうためである [6]。

## 2 コマンドラインオプション

基本的には BibTeX と同様であるが、以下が追加されている。

- `-kanji=<encoding>`  
入出力ファイルの文字コードを指定する。利用可能な *<encoding>* の値：
  - pBibTeX : euc, sjis, jis, utf8
  - upBibTeX : euc, sjis, jis, utf8, uptex
- `-kanji-internal=<encoding>`  
内部コードを指定する (upBibTeX 専用)。利用可能な *<encoding>* の値：
  - pBibTeX : なし (常に euc に固定)
  - upBibTeX : euc, uptex
- `-guess-input-enc, -no-guess-input-enc`  
入力ファイルの文字コードを推定する機能を有効/無効にする。kpathsearch ライブラリ変数 `guess_input_kanji_encoding` よりもこちらが優先される。(TeX Live 2023 以降)

## 3 kpathsearch ライブラリ変数

出力行の長さの最大を指定する変数 `max_print_line` (デフォルト値 79) が ((u)p)BibTeX では従来ハードコードされ固定されていたが、TeX Live 2023 以降 kpathsearch ライブラリの変数になった。これに伴い `texmf.cnf` の中や環境変数などで設定変更が可能になった。

入力ファイルの文字コードを推定する機能が実装され変数 `guess_input_kanji_encoding` で制御できるようになった (TeX Live 2023 以降)。pBibTeX ではデフォルト値 1 (推定する)、

---

<sup>\*8</sup> BibTeX 0.99c 以前では半角空白以外の箇所も行末に % を補って改行されたが、2010 年の 0.99d で URL などの長い文字列を壊さないように半角空白以外での分割が禁止された。

upBibTeX ではデフォルト値 0 (推定しない) となっている。

## 4 参考：jBibTeX と pBibTeX の違い

松井氏による jBibTeX 0.31 (BibTeX 0.99c ベース) から現在の pBibTeX に至った経緯は以下のとおりである。

- 1994 年, 都立大 (のち千葉大) の桜井貴文氏により, jTeX 1.6 (web2c 6.1) の配布キットに含めるための調整 → jBibTeX 0.32
- 1995 年, アスキー pTeX の配布キットに含めるための調整
- 2002 年, アスキーにより `-kanji` オプションの追加 → jBibTeX 0.33
- 2009 年, 日本語 TeX 開発コミュニティが jBibTeX をフォークし, pBibTeX に改名
- 2010 年, pBibTeX が pTeX とともに TeX Live へ収録される。のちに BibTeX 0.99d 対応
- 2012 年, upBibTeX が upTeX とともに TeX Live へ収録される。
- 2022 年, 出力 BBL ファイルでの行分割の改良 (第 1.4 節を参照)  
substring\$ 関数の改良 (第 1.1 節を参照) → pBibTeX j0.34
- int.to.chr\$, chr.to.int\$ 関数の拡張 (第 1.1 節を参照) → pBibTeX j0.35  
change.case\$ 関数の拡張 (第 1.2 節を参照) → upBibTeX u1.29  
max\_print\_line の kpathsearch ライブラリ変数化 (第 3 節を参照)  
入力ファイルの文字コードの推定機能を追加 (第 2, 3 節を参照)
- 2023 年, pBibTeX と upBibTeX のソースコード, バイナリを統合 [10]。pBibTeX は upBibTeX へのシンボリックリンクとなるが使用法は変わらない → pBibTeX j0.36

jBibTeX は当初 NTT jTeX と組み合わせて使用することを想定して開発されたため, 文字コードに関する扱いに NTT jTeX 由来のものが多かった。これらはコマンド名が `jbibtex` から `pbibtex` (pTeX と同じ接頭辞) に改名された 2009 年に削除された [4] ため, pBibTeX は jBibTeX と以下の点で異なっている:

- JIS コードにおいて日本語文字コード開始・終了を示す種々のエスケープ・シーケンス (`ESC$@` と `ESC$B`, `ESC(J` と `ESC(B` など) を区別しない。
- 環境変数 `BIBTERMCODE`, `BIBFILECODE`<sup>\*9</sup> は, pBibTeX では参照されない。

---

<sup>\*9</sup> 松井氏のドキュメント `jbibtex.pdf` [1] の「3.3 漢字コードの扱い」および `jbtxdoc.pdf` [2] の「1. 概要」の jBibTeX での注意点として言及されているもの。



## 参考文献

- [1] 松井正一, 「日本語 BibTeX : jBibTeX」, ./jbibtex.pdf
- [2] 松井正一, 「BibTeXing : BibTeX の使い方」, ./jbtxdoc.pdf
- [3] 松井正一, 「Designing BibTeX Styles — BibTeX スタイルの作り方」, ./jbtXHak.pdf
- [4] 土村展之, 「コマンド名問題 - ptexlive Wiki」  
<https://tutimura.ath.cx/ptexlive/?%A5%B3%A5%DE%A5%F3%A5%C9%CC%BE%CC%E4%C2%EA>
- [5] Haruhiko Okumura, 「upbibtex で名と姓の間のスペースが消える」, 2020/10/11,  
<https://github.com/texjporG/tex-jp-build/issues/109>
- [6] Hironobu Yamashita, 「(u)pbibtex で和文文字直後の半角スペースで改行が起きる」,  
2022/02/07,  
<https://github.com/texjporG/pbibtex-manual/issues/1>
- [7] Hironobu Yamashita, 「(u)pbibtex: freeze at en-dash etc. + sieicej.bst」, 2022/02/19,  
<https://github.com/texjporG/tex-jp-build/issues/133>
- [8] Takuji Tanaka, 「bibtexu の日本語 (CJK) 対応」, 2022/04/16,  
<https://github.com/texjporG/tex-jp-build/issues/139>
- [9] Takuji Tanaka, 「[ptexenc] 入力ファイルの文字コードの自動判定」, 2022/06/05,  
<https://github.com/texjporG/tex-jp-build/issues/142>
- [10] Takuji Tanaka, 「(u)pbibtex バイナリの整理統合」, 2022/11/30,  
<https://github.com/texjporG/tex-jp-build/issues/154>