udline.sty について

1　このパッケージの機能概要

このパッケージは，TEX で下線や飾り下線や上線 (っていうのかな)，打ち消し線を引くためのマクロを用意しています。縦書き・横書きいずれにも対応しています。なお，縦書きの場合は下線ではなく右傍線，上線ではなく左傍線ですが，以下ではまとめて「下線・上線」という言葉で呼ぶことにします。

さて，TEX には元々下線を引く機能があるのですが,「文字を強調するときは字体の変更で行うべき」という思想で作られているようで，あまり機能は高くありません。具体的に言えば，下線を引いた部分は (1 つのボックスと見なされ) 途中改行できなくなります。また，線の種類もデフォルトでは一本線だけです。

このパッケージでは，これらの問題を改善すべく,「途中改行」と「線種の拡充」を可能にします。— すでにこういった目的のパッケージファイルが多数発表されていますので，今更という感じもしますが，勢いで作っちゃったので，記念に発表します …… 何せ 7 年越しだし。(^_^;

このパッケージの特徴

- 途中改行可能な下線・飾り下線・上線・飾り上線・打ち消し線・飾り打ち消し線を引ける
- 下線・飾り下線・上線・飾り上線までの距離を調節できる
- 打ち消し線・飾り打ち消し線の位置を調節できる
- 下線・上線・打ち消し線の太さを変えられる
- 飾り線がデフォルトで 8 種用意されている
- しかも自分で新しい飾り線を定義することもできる
- 縦書き・横書き・和文・英文対応

2　このパッケージの使い方

プリアンブルに `\usepackage{udline}` と書き込んでください。その上で本文の下線等を引きたい部分に，以下のいずれかのコマンドを指定してください。たとえば この下線 は `\unc{この下線}` として生成されています。

コマンドの意味は以下の通りです。

	一本線・改行不可	飾り線・改行不可	一本線・改行可	飾り線・改行可
下　　線	`\unl{文書}`	`\unc{文書}`	`\ul{文書}`	`\uc{文書}`
上　　線	`\ovl{文書}`	`\ovc{文書}`	`\ol{文書}`	`\oc{文書}`
打ち消し線	`\sol{文書}`	`\soc{文書}`	`\Sl{文書}`	`\Sc{文書}`

(注) `\Sl` , `\Sc` の `S` は大文字です。これは小文字の `\sl` , `\sc` がすでに TEX で別の意味で定義済みのためです。なお，このパッケージのマクロは短い名前のものが多いので，もしかしたら他のパッケージのマクロ

と名前の衝突を起こすかもしれません。その場合の対処法の 1 つを，このマニュアルの最後に書いておきますので，不幸にしてそうなった場合は参考にしてください。

さて，話を戻します。マクロのうち「改行可能」の方は，TEX の行分割機能を生かすために，1 文字ずつ (英文では 1 単語ずつ) 取り出して下線等を引きます。そのため，下線等を引きたい部分に引数付きのマクロがある場合などは，その部分を保護する必要があります (後述)。これは結構面倒なので，行分割が必要ないような短い下線等の場合は,「改行不可」のコマンドを使う方が便利です。こちらは下線等を引く部分全体をひとつのボックスとしてまとめて線を引くため，内部に引数付きのマクロがあっても大丈夫です。

3　線の太さ・飾り線の種類の変更

下線・上線・打ち消し線は太さを自由に変更できます。飾り下線・飾り上線・飾り打ち消し線は (太さを自由には変更できませんが) いくつかの中から好みのものを選択できます。

まず，下線等の太さの変更ですが，これは太さを変えたい部分の `\unl` や `\ul` 等の前で，以下の「太さ変更コマンド」を指定します。

`\setulwidth{太さ}` …… 下線
`\setolwidth{太さ}` …… 上線
`\setslwidth{太さ}` …… 打ち消し線

たとえば，この段落に入る前に `\setulwidth{2pt}` と指定しているのですが，こうしておくと，以降すべて このような 太さ 2pt の下線が引けます。デフォルトは `0.1ex` ですので，もとに戻すときには `\setulwidth{0.1ex}` などと指定してください。あるいは，TEX のグルーピングの考え方を利用して，`\setulwidth` の影響の及ぶ範囲を制限することも可能です。 たとえば,

```
\unl{ここは通常}で{\setulwidth{2pt}\unl{ここは太線}}で\unl{ここはまた通常}
```

とすれば,

ここは通常でここは太線でここはまた通常

となります。

次に，飾り線の線種の変更について。以下の 8 種類が用意されています。(いずれも引数なし)

- `\setnoko` … 以降の飾り線を鋸線にします
- `\setnami` … 以降の飾り線を波線にします
- `\setNami` … 以降の飾り線を太波線にします
- `\setitten` … 以降の飾り線を一点鎖線にします
- `\setniten` … 以降の飾り線を二点鎖線にします
- `\setniju` … 以降の飾り線を二重線にします (デフォルト)
- `\sethasen` … 以降の飾り線を破線にします
- `\settensen` … 以降の飾り線を点線にします

このうち，一点鎖線・二点鎖線・破線に関しては，線種変更時に線分の長さなどを指定できるコマンドも用意しました。

- `\setItten{線の長さ}{間隙の長さ}` … 以降の飾り線を一点鎖線にします
- `\setNiten{線の長さ}{間隙の長さ}` … 以降の飾り線を二点鎖線にします
- `\setHasen[太さ]{線の長さ}{間隙の長さ}` … 以降の飾り線を破線にします

引数のうち,「線の長さ」はこれらの飾り線を構成する線分の長さ,「間隙の長さ」は一点鎖線・二点鎖線では「点」をうつ部分の長さ，破線では文字通り線分と線分の間の間隙の大きさを表します。さらに，破線ではオプション引数で「太さ」も指定できます。

これらを線種を変えたい部分の `\unc` や `\ulc` 等の前で指定すると，こんなふうに線種が替わります。一部の線種のみ変えたいときには,「太さ」の変更のときと同様にグルーピングして下さい。

なお,「改行可能タイプ」では 1 文字ずつ取り出して下線を付けるため，一点鎖線や二点鎖線や破線といった，くり返し単位の大きい下線には向いていません。一応 10〜12pt で出力したときに比較的見栄えが良くなるよう調整してはいますが，できればただの下線か二重線，点線等を使うか，あるいは「改行不可タイプ」を利用することをお薦めします。10〜12pt 以外で見栄えが良くなるようにするには，線種を定義しなおす (あるいは新規に定義する) 必要があります。線種を自分で定義する方法については，後述します。

4　改行可能タイプを使用する上での注意

すでに書きましたように，TEX 本来の下線は，途中で改行が起こることを想定していません。では「改行可能タイプ」ではどうやって改行を実現しているかというと，簡単に言えば，文字列から 1 つずつ文字を取り出し，下線等を引いて，また文字列に戻す，ということをしています。つまり，改行は TEX 本来の改行の仕組みにお任せというわけです。

しかし，この手法をもろに採用するといろいろ問題が生じます。`udline.sty` ではその問題点をある程度修正あるいは回避していますが，ユーザーの側でも若干そのことに関して理解していただくべき事項がありますので，それについて説明します。

まず，問題点を列挙します。
(最後の 2 つ以外は一応対策済み。最後の 2 つはどうしようもありません。(^_^;)

- 「。」や「,」などの禁則処理や，これらの前後の空白が無視される。
- `\frac{a}{b}` などのように引数をとるマクロでは，`\frac` と `{a}` と `{b}` に分割されてしまうためエラーになる。
- 英単語を 1 文字ずつ処理すると，合字やハイフネーションが行われなくなり，文字間が間延びしたような感じになる。
- 和文は少し文字間が詰まった感じになる。
- `Overfull` や `Underfull` の警告が出やすくなる。

順番に片づけていきますと，まず最初の禁則処理等については，プログラムである程度解決しています。禁則処理の正体は penalty の挿入ですので,「。」や「,」などの特定の記号が検出されたとき，その前や後ろに penalty を挿入しています。また，下線を引く前に文字を幅 0 の vrule ではさみ，自然な空白を入れています。完璧ではありませんが，これでかなり不自然さは解消されます。

次の引数付きのマクロについてですが，これはマクロとその引数を `{}` ではさむことによって問題を回避します。(たとえば `\ul` の引数の中に `$\frac{a}{b}$` を入れたいときは,

```
\ul{ … {$\frac{a}{b}$} … }
```

とします。) 原理は，1 文字ずつ取り出してきたとき，{ が検出されたらその括弧が閉じるまでを一気に取り出して処理する，ということです。

次に，英単語についてですが，これは英字部分を 1 文字ずつでなく，1 単語毎に抜き出すことによって問題の解決を図っています。(単語の切れ目は空白で検出) ただし，そのためには和文部分と英文部分の境目をはっきりさせる必要がありますので，英文部分を \eiji と \Eiji ではさむことにしました。ハイフネーションはさすがに無理ですので，一旦出力してみて，はみ出した部分に手動で abc-␣def のようにハイフンと空白を入れることになります。ちなみに，英文終わりを意味する \Eiji の前には必ず半角スペースを入れてください。なお，英文のみ，あるいはほとんど英文 (単語間の空白が頻繁に入る文) の場合，\eiji と \Eiji を埋め込んだ \Eul , \Euc , \Eol , \Eoc , \ESl , \ESc も使えます。

最後に文字間隔についてですが，文字に付けた線どうしがきれいにつながるようにする必要がありますので，文字間隔は文字の両側にはみ出す線の長さと関係します。文字間隔を現在より大きくするためには，はみ出す線の長さを長くすればいいのですが，そうすると，今度は行末でも線が横に大きくはみ出し不格好になります。現在の文字間隔はその辺りの妥協の産物です。

参考：「改行可能タイプ」の実例については，別ファイル sampleul.tex をご覧下さい。

5 「短縮」マクロ

波線などを引くとき，\setnami\unc{文書} とするのが面倒，という要望に応えまして，短縮マクロを用意しました。

短縮名をもつのは，\unc , \ovc , \uc , \oc , \Euc , \Eoc に対応する鋸線，波線，太波線，一点鎖線，二点鎖線，二重線，破線，点線で，上記マクロの c の部分を，それぞれ noko , nami , Nami , itten , niten , niju , hasen , tensen とした名前になっています。

たとえば，\ovnami{文章} は {\setnami\ovc{文章}} と同じ意味になります。

以下，その一覧を挙げておきます。

\unnoko	\unnami	\unNami	\unitten	\unniten	\unniju	\unhasen	\untensen
\ovnoko	\ovnami	\ovNami	\ovitten	\ovniten	\ovniju	\ovhasen	\ovtensen
\unoko	\unami	\uNami	\uitten	\uniten	\uniju	\uhasen	\utensen
\onoko	\onami	\oNami	\oitten	\oniten	\oniju	\ohasen	\otensen
\Eunoko	\Eunami	\EuNami	\Euitten	\Euniten	\Euniju	\Euhasen	\Eutensen
\Eonoko	\Eonami	\EoNami	\Eoitten	\Eoniten	\Eoniju	\Eohasen	\Eotensen

6 下線・飾り下線までの距離を変える

下線 (右傍線) と文字 (正確にはベースライン) との距離は，\setulminsep と \setulsep で指定します。前者で指定した長さを a , 後者で指定した長さを b とすると，ベースラインと下線 (右傍線) の間には $a+b$ の 距離があり，下線と次の行 (縦書きの場合は右傍線と手前の行) の間には b の空白が入ります。ただし，下線を引いた文字列の中にたとえばディスプレイスタイルの分数式のような，高さや深さの大きい「文字」があった場合はその深さ (縦書きの場合高さ) を a' として，$a'+b$ の距離がベースラインとの間隔となります。その意味で \setulminsep で指定される値は，ベースラインから下線までの距離の「最小値」を左右する値となります。

これらのマクロは次のようにして使います。

```
\setulminsep{縦書き時の寸法}{横書き時の寸法}
\setulsep{寸法}
```

縦書きのときと横書きのときでは，ベースラインの位置が違いますので，\setulminsep は縦横別々に指定することになります。ちなみにデフォルトは

```
\setulminsep{1.2ex}{0.6ex}
\setulsep{2pt}
```

です。

通常はデフォルトの値から変更する必要はないと思いますが，たとえば,「改行可能タイプ」で下線を引く部分にディスプレイスタイルの分数式を入れると，このように $\frac{a+b}{c+d}$ そこだけ下線が下がってしまいますので，そのときは \setulminsep の値を大きくしてください。また，縦書きでは右傍線とルビが同じ側につきますので，そのあたりの調整にも使えます。たとえば右の 2 つの例のように，右傍線とルビを同居させるとき，一般には傍線とベースラインとの距離を調整したいことが多いでしょう。(右の例では特に距離の調整は行っていませんが)。なお，右の 2 例は

```
こんな風に\unl{ルビと\shortstack{ぼうせん\\傍線}を同居}させたり
こんな風に\unl{ルビと\smash{\shortstack{ぼうせん\\傍線}}を同居}させる
```

として出力しています。

こんな風にルビと 傍線 を同居させる ぼうせん

こんな風にルビと ぼうせん 傍線 を同居させたり

7 上線・飾り上線までの距離を変える

上線 (左傍線) と文字 (正確にはベースライン) との距離も，下線 (右傍線) と同様に指定できます。指定に使うコマンドは \setolminsep と \setolsep で，それぞれ \setulminsep と \setulsep に対応します。使い方も下線 (右傍線) のときとほぼ同じですので，詳しくは前項を参照してください。ちなみにデフォルトは

```
\setolminsep{1.2ex}{1.8ex}
\setolsep{2pt}
```

です。

8 打ち消し線の位置を変える

打ち消し線は，線を引くボックスの「中央」に線を引きますが，ボックスの中に 1 文字だけ異様に背の高い「文字」があると，次の例のように他の文字に線がかからないことが考えられます。

(例) \sol{aoi-sora s{\huge h}iroi-kumo} で aoi-sora s$\overline{\text{h}}$iroi-kumo

この場合，\sol[a]{ ····· } とすれば，線の高さは文字「a」の中央の高さになります。(「a」自体は印字されません。)

(例) \sol[a]{aoi-sora s{\huge h}iroi-kumo} で ~~aoi-sora s~~h~~iroi-kumo~~

打ち消し線マクロ \sol ，\soc ，\Sl ，\Sc ，\ESl ，\ESc はすべてこのオプション引数をとれます。

9　飾り線の線種を自分で定義する

飾り線は「文字」をくり返して配置することによって実現しています。たとえばこの太波線は 10pt サイズのゴチックの「〜」を，5.5pt 間隔で配置することによって実現しています。これらの「くり返し単位」の指定は，\ulchar で行います。使い方は

\ulchar{使用フォント名}{使用文字 (列)}{くり返し間隔}

です。たとえば，二重線を指定する \settensen は次のように定義されています。

\def\settensen{\ulchar{cmr10}{\BLdot}{2pt}}

(\BLdot はベースライン上に中心をもつピリオド (ドット) を生成します。)

ちなみに文字に和文文字を使うときは，フォントに縦書き用と横書き用があることにご注意を。たとえば，上記の太波線を指定する \setNami は次のように定義されています。

\def\setNami{\ulchar{\iftdir tgoth10\else goth10\fi}{〜}{5.5pt}}

「使用文字 (列)」の部分は複数の文字や vrule のような罫線文字でも可能です。(\hbox{} に入れているだけですので，たいていのものは指定可能です)。

なお,「改行可能タイプ」では，前記のように 1 文字ずつ飾り下線等を引くことになりますが，その際，文字幅は飾り線のくり返し幅の整数倍に繰り上げられます。逆に言えば，飾り線のくり返し幅の整数倍が，通常使う文字の横幅よりほんのちょっと大きくなるようくり返し幅を設定すべきです。

くり返し幅の大きい飾り線では，空白に引く飾り線と通常文字に引く飾り線は変えるべきかも知れません。このときには，\ulcharC と \ulcharG を使います。前者は通常文字に引く飾り線のくり返し単位，後者は空白に引く飾り線のくり返し単位を指定します。指定方法は \ulchar と同じです。(というより \ulchar は内部で \ulcharC と \ulcharG を呼び出しているにすぎません)。

10　上線と下線を同時に引く

このパッケージには，上線と下線を同時に引く機能はありません。上線と下線を全く同じ部分に引くなら，たとえば

\ovnoko{\unNami{夕日が沈む}}　　で　　夕日が沈む

となりますが，右のように引く部分が違うとどうしようもありません。　　赤い夕日が沈む

えっ? 引けてるって? いえいえ，これは「ごまかし」ているだけです。ここではちょっとごまかしのテクニックを述べましょう。

まず，\makebox[0mm][l]{\ovnoko{赤い夕日}} とします。これは「赤い夕日」に上線 (鋸線) を引き，その幅を 0 にしています。したがって，この後に続く文は，この「赤い夕日」に重ねて印字されます。

もうおわかりでしょうか。重ね印字する文は「赤い」に相当する空白 \hspace{2zw} (上線を引いたため厳密に 2zw ではない。試行錯誤で調整) と，下線 (太波線) を引いた「夕日が沈む」です。ただし,「夕

日」の部分が微妙にずれると印字した際文字が太くなってしまいますので，必要なら \phantom で見えなくしてしまうといいでしょう。

結局，上の文は以下のようにして生成されました。

```
\makebox[0mm][l]{\ovnoko{赤い夕日}}\hspace{2zw}{\unNami{\phantom{夕日}が沈む}}
```

11　他のパッケージとの衝突回避

あるマクロパッケージを単独で使用したときは問題なかったのに，別のパッケージを読み込んだら動かなくなった，ということはままあることです。原因としては，マクロや変数の衝突が考えられます。

\unitlength のような変数の名前が衝突しているのであれば，マクロを使用する際にその変数に適当な値を設定し，その部分をグルーピング ({ } でくくる) することで何とかなりますし，\input で読み込むマクロも，グルーピングで他の部分に影響を与えないようにできます。しかし，\usepackage を用いて読み込むマクロにはこの方法は使えません。以下ではそういった場合の対処法の一例を述べます。

対処法のアウトラインは,「衝突しているマクロを一旦テンポラリーマクロ (?) に代入 (保存) しておき，切り替えマクロでそれを復活させる」というものです。これなら，衝突が起こったとき，両方のパッケージのソースを変更することなく個別対応ができます。

たとえば，MyMacro.sty というパッケージと，udline.sty で \ol というコマンドに異なる定義を与えていたとします。このとき，次のようにすれば衝突を回避できます。

まず，次のような 2 つのファイルを作っておきます。

```
%=========preul.tex===================================
\makeatletter
\let\Myol=\ol%                \ol を \Myol の名前で保存
\makeatother
\endinput
%======================================================

%=========postul.tex==================================
\makeatletter
\let\ovline=\ol%            \ol を \ovline の名前で保存
\def\UseUdline{\let\ol=\ovline}%        切り替えマクロ
\def\UseMyMacro{\let\ol=\Myol}%                  の定義
\makeatother
\endinput
%======================================================
```

この 2 つのファイルを，スタイルファイルと一緒のところに保存しておき，使うときは次の順序で読み込みます。

```
%======================================================
\usepackage{MyMacro}%     (step1)
```

```
\input{preul}%              (step2)
\usepackage{udline}%        (step3)
\input{postul}%             (step4)
%=====================================================
```

各ステップで行われていることは以下の通りです。
(step1)MyMacro.sty を読み込みます。これによって MyMacro.sty 流 \ol が定義されます。
(step2)preul.tex を読み込みます。これによって MyMacro.sty 流 \ol に \Myol という別名が与えられます。
(step3)udline.sty を読み込みます。これによって \ol の定義が udline.sty 流に上書きされます。しかし，\Myol の定義は元のままです。
(step4)postul.tex を読み込みます。これによって udline.sty 流 \ol に \ovline という別名が与えられ，MyMacro.sty 流 と udline.sty 流 の 2 つの定義を切り替えるマクロも定義されています。

したがって，以降では，2 つの \ol を混ぜて使いたいときには \Myol , \ovline を使い，過去に作った文書をそのまま使うときは，切り替えマクロ \UseMyMacro ，\UseUdline で \ol の意味を切り替えて使えます。

この対処法は，スタイルファイルそのものに手を加えることがないので，柔軟性という点では一番妥当な選択なのですが，それでも，どのマクロが衝突しているか見極める力が必要となりますので，そう易しいものではありません。(プリアンブルに \setcounter{errorcontextlines}{8} を入れてタイプセットすると，エラーメッセージが詳しくなり，衝突しているマクロが特定しやすくなります。errorcontextlines は最高何段階までエラー箇所を展開するかを指定する変数で，8 段階くらい指定しておけば十分でしょう。)