Specialized Emacs Features(日本語訳)

このマニュアルは、Emacs の特別な機能について説明します。

Table of Contents

イントロダクション

このマニュアルには、印刷された Emacs マニュアルに含めるには特化的に過ぎる、さまざまな機能についての詳細な情報が含まれています。このマニュアルは、Emacs の基本的な知識をもつ人が読めるように意図されています。しかし、いくつかのセクションは、Elisp 作者のような、より専門的な読者を意図しているものもあります。これについては、それらのセクションの冒頭で明確に指摘されているはずです。

特定のパッケージ (または関連する機能のコレクション) には、メインの Emacs マニュアルとは別に、自身のマニュアルがあります。このマニュアルは、それらの追加のマニュアルを読むことにたいする代替ではなく、補足であることを意図しています。一言で言うと、このマニュアルは、自身のマニュアルをもったり印刷された Emacs マニュアルに含めたりするには、あまりに些細な (あるいは、あまりにマイナーな)、小さな特定の機能 (または標準機能についての追加詳細) のコレクションです。しかし、このマニュアルのチャプターは、Info ドキュメントとしてフォーマットされたメインの Emacs マニュアルには、(関連する場所に) 含まれています。

特に Elisp プログラマーを意図したセクションは、Elisp マニュアルのスタイルにしたがうことができます。その他のセクションは、Emacs マニュアルのスタイルにしたがうでしょう。

ピクチャーの編集

テキスト文字で作られたピクチャー (たとえばプログラムのコメントにあるような、レジスターをフィールドに分割したピクチャーなど) を編集するには、コマンド *M-x picture-mode*を使用して Picture モードに入ります。

Picture モードでは、編集はテキストの *quarter-plane(1/4* 平面*)* モデルにもとづき、テキスト文字は、右方および下方に無限に広がる領域に点在します。このモデルには、行の終端という概念は存在しません。せいぜい言えるのは、その行の空白でない最後の文字がどこにあるかぐらいです。

もちろん、実際には Emacs はテキストを文字シーケンスとして考え、行は終端をもちます。しかし Picture モードは、もっともよく使われるコマンドを、テキストの quarter-plane モデルをシミュレートする変種に置き換えます。これらのコマンドはスペースの挿入、またはタブをスペースに置き換えることにより、これを行ないます。

Emacs のほとんどの基本的な編集コマンドは、本質的には同等なことを quarter-plane の方法で行なうように、Picture モードにより再定義されます。それに加えて Picture モードは、*C-c*プレフィクスで始まる、特別なピクチャー編集コマンドを実行する、さまざまなキーを定義します。

これらのキーのうち特に重要なのは *C-c C-c*です。ピクチャーが、通常は他のメジャーモードで編集する、大きなファイルの一部ということもあります。Picture モードは以前のメジャーモードを記録するので、後で *C-c C-c* (`picture-mode-exit`) コマンドでそのモードに戻ることができます。*C-c C-c*は数引数を与えなければ、行末のスペースの削除も行ないます。

Picture モードの特別なコマンドのすべては、(`picture`ライブラリーがロードされていれば) 他のモードでも機能しますが、Picture モード以外ではキーにバインドされません。以下の説明では、“1 列” 移動... のような言い方をしますが、通常の同等なコマンドと同じように、Picture モードのコマンドは数引数を扱うことができます。

Picture モードをオンにすることにより、フック `picture-mode-hook`が実行されます。Picture モードにたいする追加の拡張は、`artist.el`で見ることができます。

Picture モードでの基本的な編集

ほとんどのキーは、Picture モードでも通常と同じことを、quarter-plane スタイルで行ないます。たとえば *C-f*は、`picture-forward-column`を実行するようにリバインドされます。これはポイントを 1 列右に移動します。必要ならスペースを挿入するので実際の行末は関係ありません。*C-b* は、`picture-backward-column`を実行するようにリバインドされます。必要ならタブを複数のスペースに変換して、常に 1 列ポイントを左に移動します。*C-n*と *C-p*は、`picture-move-down`と `picture-move-up`を実行するようにリバインドされ、どちらもポイントが同じ列に留まるように、必要に応じてスペースの挿入とタブの変換を行ないます。*C-e*は、`picture-end-of-line`を実行します。これは、その行の最後の非空白文字の後に移動します。*C-a*は、`picture-beginning-of-line` を実行します (スクリーンモデルの選択は行の開始に影響を与えません。このコマンドが行なう追加事項は、カレントピクチャー列を 0 に更新することです)。

テキストの挿入は、Overwrite モードを通じて quarter-plane モデルに適合されます (Section “Minor Modes” in *the Emacs Manual* を参照してください)。自己挿入文字は既存のテキストを右にずらすのではなく、列ごとに既存のテキストを置き換えます。`RET`は `picture-newline`を実行し、これは単に次の行の先頭に移動するので、新しいテキストでその行を置き換えることができます。

Picture モードでは、通常は削除や kill を行なうコマンドは、かわりにテキストを消去 (スペースで置き換え) します。`DEL` (`picture-backward-clear-column`) は、前の文字を削除するのではなく、スペースで置き換えます。これはポイントを後方に移動します。*C-d* (`picture-clear-column`)

は、次の文字をスペースで置き換えますが、ポイントは移動しません (文字をクリアーして前方に移動したいときはSPCを使用します)。*C-k* (picture-clear-line) は、実際に行の内容を kill しますが、バッファーから改行は削除しません。

実際に挿入を行なうには、特別なコマンドを使わなければなりません。*C-o* (picture-open-line) は、カレント行の後に空行を作成します。行の分割はしません。Picture モードでも *C-M-o* (split-line) は意味があるので、変更されていません。*C-j* (picture-duplicate-line) は、カレント行の下に同じ内容の行を挿入します。

Picture モードで実際の削除を行なうには、*C-w*、*C-c C-d*(これは他のモードでは *C-d*にバインドされている delete-charにバインドされています)、またはピクチャー矩形コマンドの 1 つを使用します ([Rectangles in Picture], page 4 を参照してください)。

挿入後の移動の制御

Picture モードでは、“自己挿入” 文字は上書きとポイント移動を行なうので、ポイントを移動する方法に、本質的に制限はありません。ポイントは通常右に移動しますが、“自己挿入” 文字の後に移動する方向は、直行方向と対角方向の 8 つのうちから任意の方向を指定できます。これはバッファーに線を描くとき便利です。

C-c <
C-c LEFT 挿入の後、左に移動します (picture-movement-left)。

C-c >
C-c RIGHT 挿入の後、右に移動します (picture-movement-right)。

C-c ^
C-c UP 挿入の後、上に移動します (picture-movement-up)。

C-c .
C-c DOWN 挿入の後、下に移動します (picture-movement-down)。

C-c ‘
C-c Home 挿入の後、左上 (“北西”) に移動します (picture-movement-nw)。

C-c ’
C-c prior 挿入の後、右上 (“北東”) に移動します (picture-movement-ne)。

C-c /
C-c End 挿入の後、左下 (“南西”) に移動します (picture-movement-sw)。

*C-c *
C-c next 挿入の後、右下 (“南東”) に移動します (picture-movement-se)。

カレント Picture 挿入方向にもとづき移動を行なうコマンドは 2 つです。1 つはコマンド *C-c C-f* (picture-motion) で、その時点で “挿入” 後に移動すると定められた方向と同じ方向に移動するのが *C-c C-f* (picture-motion) で、逆方向へは *C-c C-b* (picture-motion-reverse) です。

Picture モードでのタブ

Picture モードでは、タブのような動作が 2 つ提供されます。コンテキストベースのタブ動作には、*M-TAB* (picture-tab-search) を使用します。引数を与えないと、前の空でない行で、空白の後の次の “意味をもつ” 文字の下にポイントを移動します。ここで “次” とは、“開始した位置から水平方向に大な位置” という意味です。*C-u M-TAB*のように引数を指定した場合、このコマンドはカレント

行で次の意味のある文字に移動します。*M-TAB*はテキストを変更せず、ポイントだけを移動します。“意味のある” 文字は変数 `picture-tab-chars`により定義され、これは一連の文字で定義されます。この変数の構文は正規表現での‘`[...]`’の内側で使用される構文と似ていますが、‘`[`’と‘`]`’はありません。デフォルト値は`"!-~"`です。

`TAB`は`picture-tab`を実行し、これはカレントのタブストップの設定にもとづき動作します。Picture モードでは`tab-to-tab-stop`と等価です。通常はポイントを移動するだけですが、数引数を指定した場合は、移動先までのテキストをクリアーします。

コンテキストベースとタブストップベースのタブ動作形式は、*C-c TAB* (`picture-set-tab-stops`) で合わせることができます。このコマンドは、カレント行にたいして、*M-TAB*が意味をもつと判断するであろう位置に、タブストップをセットします。このコマンドの使い方としては、`TAB`と合わせて、コンテキストベースの効果を得ることができます。しかし *M-TAB*で充分な場合は、そちらのほうが便利です。

ピクチャー内では、実際のタブ文字を抑止するほうが便利かもしれません。たとえば、これにより *C-x TAB*がピクチャーをめちゃくちゃにするのを防ぐことができます。変数 `indent-tabs-mode`を`nil`にセットすることにより、これを行なうことができます。

Picture モードの矩形領域コマンド

Picture モードは、quarter-plane モデルに適合する方法で、テキストの矩形部分に作用するコマンドを定義します。標準の矩形領域コマンドも便利でしょう Section “Rectangles” in *the Emacs Manual* を参照してください。

C-c C-k
: 矩形領域をスペースでクリアーします (`picture-clear-rectangle`)。プレフィクス引数を指定した場合、テキストを削除します。

C-c C-w r
: 同様ですが、最初にレジスター *r*に矩形領域の内容を保存します (`picture-clear-rectangle-to-register`)。

C-c C-y
: ポイント位置を左上隅として、最後に kill された矩形領域をバッファーに上書きコピーします。引数を指定した場合は、上書きではなく挿入します。

C-c C-x r
: 同様ですが、レジスター *r*の矩形領域を使用します (`picture-yank-rectangle-from-register`)。

ピクチャー矩形領域コマンドの *C-c C-k* (`picture-clear-rectangle`) と *C-c C-w* (`picture-clear-rectangle-to-register`) が、標準の矩形領域コマンドと異なる点は、通常は矩形領域を削除するかわりにクリアーすることです。これは Picture モードで *C-d*が変更された方法と類似しています。

しかし Picture モードで矩形領域を削除するのが便利なときもあるかもしれないので、これらのコマンドは数引数を与えたときは矩形領域を削除します。数引数の指定の有無に関わらず、*C-c C-k* は *C-c C-y*のために矩形領域を保存します。

矩形領域を yank する Picture モードのコマンドは、挿入ではなく上書きするという点で、標準の矩形領域コマンドと異なります。これは Picture モードでのテキストの挿入方法が他のモードと異なるのと同じです。*C-c C-y* (`picture-yank-rectangle`) は、一番最近 kill された矩形領域を (上書きにより) 挿入し、*C-c C-x* (`picture-yank-rectangle-from-register`) は、指定されたレジスターの矩形領域で同様のことを行ないます。

非ファイルバッファーの自動リバート

通常 Global Auto Revert モードは、ファイルのバッファーだけをリバートします。非ファイルバッファーにたいして自動リバートを行うには、2 つの方法があります。1 つはそれらのバッファーにたいして Auto Revert モードを有効にする方法です (*M-x auto-revert-mode*を使います)。もう 1 つは `global-auto-revert-non-file-buffers`に非 `nil`値をセットする方法です。後者はそれが実装されているすべての種類のバッファーにたいして、自動リバートを有効にします (以下のメニューにリストされています)。

ファイルバッファーと同様、非ファイルバッファーはそれらにたいして作業を行っているときや、リバートすると失われてしまう情報が含まれているとき、通常はリバートすべきではありません。したがって、それらが “modified(変更されている)” のときはリバートしません。非ファイルバッファーが変更されているとマークするのは、ファイルバッファーのときより通常難しいので、トリッキーになり得ます。

他のトリッキーな点に関する詳細は、効率の問題です。自動リバートはしばしばバッファーにたいするすべての可能な変更を検知しようとせず、“広範” または簡単に検知できる変更だけを検知します。したがって、非ファイルバッファーに自動リバートを有効にすることは、バッファーのすべての情報が最新であると常に保証はしませんし、手動によるリバートを無用にする必要もないからです。

それとは反対に、特定のバッファーは `auto-revert-interval`で指定された秒数ごとに自動的にリバートします (これは現在のところ Buffer Menu だけに適用されます)。この場合、自動リバートはリバートの際、`auto-revert-verbose`が非 `nil`でも、何もメッセージを表示しません。

詳細はバッファーの特性に依存し、それらは対応するセクションで説明されています。

Buffer Menu の自動リバート

非ファイルバッファーの自動リバートが有効なとき、必要性の有無にかかわらず、Buffer Menu は `auto-revert-interval`で指定された秒数ごとに自動的にリバートされます。(これは多分実際にリバートする必要があるかチェックするより長い間隔です)。

Buffer Menu が modified と不適切にマークされたときは、*g*により手動でリバートして自動リバートを再開します。しかし特定のバッファーにたいして削除や表示のマークをつけた場合は、慎重になる必要があります。なぜならリバートはすべてのマークを消去するからです。マークの追加はバッファーの modified フラグをセットするという事実は、自動的なマークの消去からの自動リバートを防ぎます。

Dired バッファーの自動リバート

Dired バッファーの自動リバートは、現在のところ Unix スタイルのオペレーティングシステムだけで機能します。他のシステムでは満足に機能しないでしょう。

Dired バッファーは、バッファーのメインディレクトリーのファイルリストが変化したとき (たとえば新しいファイルの追加) だけ、自動リバートします。特定のファイルの情報が変化したとき (たとえばサイズの変化) や、サブディレクトリーへの追加は自動リバートしません。すべてのリストされた情報が最新であることを確実にするには、Dired バッファーで自動リバートが有効であっても、手動で *g*を使ってリバートする必要があります。メインディレクトリーにリストされているファイルへの変更や保存で、実際に自動リバートが起こるのに気付くかもしれません。これはファイルの変更や保存は、たとえばバックアップファイルや auto-save ファイルにより、高い確率でディレクトリー自身を変更するからです。しかし、これは保証されているわけではありません。

Dired バッファーが modified とマークされ、残したい変更がない場合、大抵は *g*でバッファーを手動でリバートすることにより、自動リバートを再開できます。しかし 1 つ例外があります。ファイルにフラグやマークをつけた場合、安全にバッファーをリバートできます。これはフラグやマークを消去しません (もちろんマークされたファイルが削除されていない場合です)。しかしバッファーはリバートの後でさえ modified の状態に留まるので、自動リバートは再開しません。これは、もしファイルにたいしてフラグやマークをつけた場合、それはバッファーにたいして作業をしており、警告なしにバッファーが変更されることを望まないだろうからです。マークやフラグがある状態で自動リバートを再開したいときは、*M-~*を使ってバッファーを non-modified とマークします。しかしマークやフラグの追加・削除・変更により、バッファーは再び modified とマークされます。

リモートの Dired バッファーは、自動リバートされません (これは遅くなるからです)。特定のファイルだけをリストするように、シェルのワイルドカードやファイル引数を与えられた Dired バッファーも同じです。`*Find*`および`*Locate*`は、どちらも自動リバートしません。

追加のバッファーにたいする自動リバートのサポートの追加

このセクションは、新しいタイプのバッファーに自動リバートのサポートを追加したい Elisp プログラマーのために意図されています。

バッファーに自動リバートのサポートを追加するには、まず最初に適切な `revert-buffer-function`をもつことです。See Section “Reverting” in *the Emacs Lisp Reference Manual*. を参照してください。

それに加えて、適切な `buffer-stale-function`をもたなければなりません。

`buffer-stale-function` [Variable]

この変数の値は、バッファーがリバートする必要があるかチェックする関数です。これは 1 つのオプション引数 *noconfirm*をもつ関数です。この関数はバッファーがリバートされるべきときは、非 `nil`を返します。バッファーは、この関数が呼ばれたときのバッファーです。

この関数は主に自動リバートで使うことを意図していますが、他の目的にも同様に使うことができます。たとえば自動リバートが無効の場合、これはバッファーをリバートする必要があるとユーザーに警告するのに使えます。このアイデアは引数 *noconfirm*が背景にあります。もしこれが `t`のときは、ユーザーに尋ねることなくリバートを行い、この関数をバッファーが無効だとユーザーに警告するために使うときは、`nil`を指定します。特に自動リバートに使う場合、*noconfirm*は `t`になります。関数を自動リバートだけに使う場合、引数 *noconfirm*は無視できます。

(Buffer Menu のように)`auto-revert-interval`で指定した秒数ごとに自動リバートを自動的に行いたいだけのときは、以下のようにします:

```
(setq-local buffer-stale-function
    #'(lambda (&optional noconfirm) 'fast))
```

これをバッファーのモード関数の中に記述します。

特別な戻り値 ‘`fast`’は、呼出側にリバートの必要性チェックは必要ないが、バッファーのリバートは早く行うことを呼出側に伝えます。これは `auto-revert-verbose`が非 `nil`のときも、自動リバートがリバートメッセージを表示しないことも伝えます。これは重要です。なぜなら `auto-revert-interval`秒ごとにリバートメッセージが表示されるのは、とても煩わしくなり得るからです。この戻り値で提供される情報は、関数が自動リバート以外の目的で使われる場合にも有用です。

バッファーが適切な`revert-buffer-function`および`buffer-stale-function`をもっていても、通常は問題がいくつか残ります。

バッファーは、unmodified とマークされているときだけ自動リバートします。したがって、さまざまな関数がバッファーを modified とマークするのは、バッファーがリバートされることにより失われる情報を持つとき、またはユーザーがバッファーで作業しているので、自動リバートが迷惑だと信じる理由があるときだと確信する必要があります。ユーザーは modified のステータスを手作業で調整することにより、常にこれをオーバーライドできます。これをサポートするために、unmodified とマークされたバッファーでの`revert-buffer-function`の呼び出しは、そのバッファーの unmodified のマークを維持すべきです。

一連の自動リバートで、ポイントがあちこちジャンプしないことを保証するのは重要です。もちろんバッファーが活発に変化するとき、ポイントが移動するのは仕方ありません。

`revert-buffer-function`が、`auto-revert-verbose`が`t`のとき出力される自動リバート自身のメッセージと重複する、不要なメッセージを出力しないことと、`auto-revert-verbose`にセットされた`nil`値を、効果的にオーバーライドすることを確実にするべきです。したがってモードを自動リバートに適合させることは、しばしばそのようなメッセージの削除をともないます。これは`auto-revert-interval`秒ごとに自動リバートを行うバッファーで特に重要です。

新しい自動リバートが Emacs の一部となったときは、`global-auto-revert-non-file-buffers`のドキュメント文字列にそれを記載するべきです。

同様にこのチャプターにセクションを追加するべきです。そのセクションは、バッファーの自動更新を有効にすることによりバッファーのすべての情報が完全に最新 (または最新から`auto-revert-interval`秒後) であると保証されるかを、最小限かつ明確にすべきです。

Dired でのサブディレクトリーにたいするスイッチ

*C-u i*を使うことにより、lsのスイッチを指定して、Dired バッファーにサブディレクトリーを挿入できます。すでに挿入されたサブディレクトリーの lsスイッチを変更するには、その位置で *C-u l*を使用します。

バッファーをリバートしたとき、Dired はこれらのスイッチを保持します。サブディレクトリーを削除すると、それにたいするスイッチは忘れられます。

dired-undo (Section “Marks vs Flags” in *the Emacs Manual* を参照してください) を使用することにより、スイッチを明示的に指定して挿入されたサブディレクトリーにたいして、それらのスイッチを記憶する (または忘れる) ための Dired の機構をバイパスして、それらのサブディレクトリーの再挿入または削除を行なうことができます。dired-undoを使用してサブディレクトリーを削除しても、それのスイッチは記憶されています。後で *i*を使ってサブディレクトリーを再挿入すると、これは古いスイッチを使用してサブディレクトリーを再挿入します。(dired-undoではなく) 通常の Dired コマンドを使って削除されたサブディレクトリーを、dired-undoを使って再挿入すると、最初は古いスイッチを使ってサブディレクトリーを挿入します。しかしバッファーをリバートすると。バッファーのデフォルトのスイッチを使って再リストします。これが問題な場合、*C-u i*か *C-u l*で、この状況を正すことができます。

Dired は Rスイッチを記憶しません。Rスイッチを含むスイッチでサブディレクトリーを挿入すると、それはそのサブディレクトリーの各サブディレクトリーそれぞれにたいして、残りのスイッチを使用して挿入を行なうのと等価です。たとえば Rで挿入されたサブディレクトリを更新または kill しても、それのサブディレクトリーは更新または kill されません。

バッファーのデフォルトスイッチは、明示的にスイッチを指定して挿入されたサブディレクトリーに影響を与えません。特にバッファーのスイッチを変更する *s*のようなコマンドは、そのようなサブディレクトリーに影響を与えません (しかし明示的にスイッチを割り当てられていないサブディレクトリーには影響します)。

*M-x dired-reset-subdir-switches*を使うことにより、Dired にサブディレクトリーにたいするスイッチをすべて忘れさせて、すべてのサブディレクトリーを再びリストさせることができます。これは Dired のバッファーもリバートします。

カレンダーとダイアリーの上級な機能

このセクションでは、カレンダーとダイアリーの、より上級で特別な機能をいくつか説明します。まず最初に、個人的な好みに合うようにカレンダーをカスタマイズする、多くの方法を紹介します。

カレンダーのカスタマイズ

残念ながら、カレンダーの表示が 3ヶ月であるのを変更することはできませんが、変数 calendar-left-margin、calendar-day-header-width、calendar-day-digit-width、calendar-column-width、calendar-intermonth-spacingをカスタマイズすることにより、使用される空白文字をカスタマイズすることはできます。各月の間に、たとえば週の番号を表示するには、変数 calendar-intermonth-headerと calendar-intermonth-textを、変数のドキュメントに記載されているようにカスタマイズします。

変数 calendar-month-headerは、カレンダーの各月の上に表示されるテキストを制御します。デフォルトでは月と年を表示します。変数 calendar-day-header-arrayは、各月の各曜日の上に表示されるテキストを制御します。デフォルトでは、各曜日の名前の最初の 2 文字を表示します。

変数 calendar-holiday-markerは、休日をどのようにマークするか指定します。この変数は、日付の隣に挿入する 1 文字の文字列か、その日付を表示するのに使用するフェイス名です。同様に、変数 diary-entry-markerは、ダイアリーエントリーをもつ日を、どのようにマークするか指定します。関数 calendar-mark-todayは、今日の日付をマークするのに calendar-today-markerを使用します。デフォルトでは、この目的のためにカレンダーは holiday、diary、calendar-todayという名前のフェイスを使います。

変数 calendar-load-hookは、calendar パッケージが最初にロードされたとき (実際にカレンダーの表示が開始される前) に実行されるノーマルフックです。

カレンダーの開始により、ノーマルフック calendar-initial-window-hookが実行されます。カレンダー表示の再計算では、このフックは実行されません。しかし *q*でカレンダーを離れ、再度カレンダーに入ると、このフックが再び実行されます。

変数 calendar-today-visible-hookは、カレンダーバッファーがカレンダーのために準備された後、カレント日付がウィンドウで表示されるときに実行されるノーマルフックです。このフックの用途の 1 つは、今日の日付のマークです。これを行なうには、関数 calendar-mark-todayまたは calendar-star-dateを使います:

```
(add-hook 'calendar-today-visible-hook 'calendar-mark-today)
```

同様のノーマルフック calendar-today-invisible-hookは、カレント日付がウィンドウに表示されなくなるとき実行されます。

カレンダーのカーソル移動コマンドは、カーソルを移動した後に、フック calendar-move-hookを実行します。

休日のカスタマイズ

Emacs が知る、デフォルト休日のリスト変数がいくつかあります。それは holiday-general-holidays、holiday-local-holidays、holiday-solar-holidays、holiday-bahai-holidays、holiday-christian-holidays、holiday-hebrew-holidays、holiday-islamic-holidays、holiday-oriental-holidays、holiday-other-holidaysです。変数の名前は自己説明的であるべきです。たとえば、holiday-solar-holidaysは、太陽と月に関連した休日のリストです。

これらの休日リストにたいして、必要に応じて、下記で説明しているように休日を削除したり追加してカスタマイズできます。これらを `nil`にセットすると、関連する休日は表示されなくなります。

一般的な休日、すなわち holiday-general-holidays は、デフォルトでは United States で一般的な休日です。対照的に、`holiday-local-holidays`と `holiday-other-holidays`は、デフォルトでは空です。前者はシステムワイドなセッティング、後者は個人的な使用を意図しています。

デフォルトでは、Emacs は世俗的なカレンダーに一般的に見出されるものを除き、宗教的な休日のすべてを含んではいません。宗教的な休日の、より広範なコレクションのために、変数 `calendar-bahai-all-holidays-flag`、`calendar-christian-all-holidays-flag`、`calendar-hebrew-all-holidays-flag`。`calendar-islamic-all-holidays-flag`のどれか (またはすべて) を `t`にセットできます。

それぞれの休日の変数は *holiday forms* のリストです。各 form は休日 (休日のリストの場合もある) を記述します。以下は利用可能な holiday form の表です。日付と月は 1 から数えますが、“曜日名 (dayname)” は日曜日を 0 として数えます。引数 *string*は、その休日を説明する文字列です。

`(holiday-fixed` *`month day string`*`)`
: グレゴリオ暦 (Gregorian calendar) の固定日付です。

`(holiday-float` *`month dayname k string`* `&optional` *`day`*`)`
: グレゴリオ暦の *month*月 *day*日の前または後の *k*番目の曜日 *dayname*(*dayname*=0 の場合は日曜日) です。*k*が負の場合、月の最後から数えます。オプションの *day*のデフォルトは、*k*が正のときは 1、負のときは *month*の最後の日になります。

`(holiday-chinese` *`month day string`*`)`
: 旧暦 (Chinese calendar) の固定日付です。

`(holiday-hebrew` *`month day string`*`)`
: ヘブライ暦 (Hebrew calendar) の固定日付です。

`(holiday-islamic` *`month day string`*`)`
: イスラム暦 (Islamic calendar) の固定日付です。

`(holiday-julian` *`month day string`*`)`
: ユリウス暦 (Julian calendar) の固定日付です。

`(holiday-sexp` *`sexp string`*`)`
: Lisp 式 *sexp*により計算される日付です。式は計算に変数 `year`を使い、`(`*`month day year`*`)`の形式のリストを返すか、その年に休日が発生しない場合は `nil`を返すべきです。

`(if` *`condition holiday-form`*`)`
: 条件 *condition*が真のときだけ休日が発生します。

`(`*`function`* [*`args`*]`)`
: 引数 *args*を指定して関数 *function*を呼び出すことにより計算される日付のリストです。

たとえばフランスで有名な 7 月 14 日のパリ革命記念日 (Bastille Day) を追加したいとしましょう。以下でこれを行なうことができます:

```
(setq holiday-other-holidays '((holiday-fixed 7 14 "Bastille Day")))
```

多くの休日は、特定の月の特定の曜日に発生します。以下は Virgin Islands で有名な 8 月第 4 月曜日の Hurricane Supplication Day を記述します。

```
(holiday-float 8 1 4 "Hurricane Supplication Day")
```

ここで 8 は 8 月、1 は月曜日 (日曜日は 0、火曜日は 2 です)、4 はその月の 4 番目 (1 は 1 番目、2 は 2 番目、−1 は最後、−2 は最後から 2 番目) を指定しています。

Bah^^c3^^a1'^^c3^^ad 暦、旧暦、ヘブライ暦、イスラム暦、ユリウス暦の固定日付に発生する休日を指定することもできます。たとえば、

```
(setq holiday-other-holidays
      '((holiday-hebrew 10 2 "Last day of Hanukkah")
        (holiday-islamic 3 12 "Mohammed's Birthday")
        (holiday-julian 4 2 "Jefferson's Birthday")))
```

これは、Hanukkah の最後の日 (ヘブライ暦の月は Nisan を 1 として数えられます)、Mohammed の誕生日を祝うイスラムの祭日 (イスラム暦の月は Muharram を 1 として数えられます)、そしてユリウス暦の 1743 年 4 月 2 日の Thomas Jefferson の誕生日を記述したものです。

条件付きの休日を含めるには、Emacs Lisp の `if`か、`holiday-sexp`形式を使用します。たとえばアメリカ大統領選挙は、4 で割りきれる年の 11 月の第 1 月曜日の後の、最初の火曜日に発生します:

```
(holiday-sexp '(if (zerop (% year 4))
                   (calendar-gregorian-from-absolute
                    (1+ (calendar-dayname-on-or-before
                         1 (+ 6 (calendar-absolute-from-gregorian
                                 (list 11 1 year)))))))
              "US Presidential Election")
```

または

```
(if (zerop (% displayed-year 4))
    (holiday-fixed 11
           (calendar-extract-day
             (calendar-gregorian-from-absolute
               (1+ (calendar-dayname-on-or-before
                     1 (+ 6 (calendar-absolute-from-gregorian
                             (list 11 1 displayed-year)))))))
           "US Presidential Election"))
```

休日の決定に特別な計算が含まれるために、上記の形式に当てはまらない休日もあります。そのような場合は、その計算を行なう Lisp 関数を記述しなければなりません。たとえば食 (eclipses) を含めるには `holiday-other-holidays`に `(eclipses)`を追加して、以下のような、カレンダーウィンドウに表示されている月に関連するグレゴリオ暦の日付のリスト (空の場合もあり得る) を返す、Emacs Lisp 関数 `(eclipses)`を記述します。

```
(((6 4 2012) "Lunar Eclipse") ((11 13 2012) "Solar Eclipse") ... )
```

マヤ暦からの変換

以下は、マヤ暦 (Mayan calendar) にもとづく日付を選択するコマンドです:

g m l long count カレンダーで指定された日付に移動します (`calendar-mayan-goto-long-count-date`)。

g m n t tzolkin カレンダーの次の周期の日付に移動します (`calendar-mayan-next-tzolkin-date`)。

g m p t tzolkin カレンダーの前の周期の日付に移動します (`calendar-mayan-previous-tzolkin-date`)。

g m n h haab カレンダーの次の周期の日付に移動します (`calendar-mayan-next-haab-date`)。

g m p h haab カレンダーの前の周期の日付に移動します (`calendar-mayan-previous-haab-date`)。

g m n c マヤ暦の次の周期の日付に移動します (`calendar-mayan-next-calendar-round-date`)。

g m p c マヤ暦の前の周期の日付に移動します (`calendar-mayan-previous-calendar-round-date`)。

これらのコマンドを理解するためには、マヤ暦 (Mayan calendars) を理解する必要があります。*long count* は以下の単位にもとづいて日数を計算します:

1 kin = 1 日　1 uinal = 20 kin　1 tun = 18 uinal
1 katun = 20 tun　1 baktun = 20 katun

したがって long count の日付 12.16.11.16.6 は、12baktun の 16katun の 11tun の 16uinal の 6kin を意味します。Emacs のカレンダーはマヤ暦の long count を 7.17.18.13.3 まで遡ることができます。*g m l*コマンドを使うときはマヤ暦の long count の日付 baktun、katun、tun、uinal、kin をピリオドで区切って入力します。

マヤ暦の tzolkin は、13 日と 20 日の独立した周期を組み合わせた形式からなる 260 日周期のカレンダーです。この周期が永遠に繰り替えされるので、Emacs はこのサイクルの前または次の位置に、後方または前方に移動するコマンドを提供します。*g m p t*は前の tzolkin 日付に移動します。Emacs は tzolkin 日付の入力を求め、前の周期のその日付にポイントを移動します。同様に、*g m n t*とタイプすると、次の周期の tzolkin 日付に移動します。

マヤ暦の haab は、20 日からなる 18 の月と、月に属さない 5 日間からなる 365 日周期のカレンダーです。tzolkin の周期と同様に、この周期は永遠に繰り返されるので、この周期の前または次の位置へ後方または前方に移動するコマンドがあります。*g m p h*は前の haab 日付に移動します。Emacs は haab 日付の入力を求め、前の周期のその日付にポイントを移動します。同様に、*g m n h*とタイプすると次の周期の haab 日付に移動します。

マヤ暦では tzolkin 日付と haab 日付を組み合わせた日付も使用されていました。この組み合わせや *calendar round* と呼ばれる、約 52 年の周期です。*g m p c*とタイプすると、Emacs は haab 日付と tzolkin 日付の入力を求め、前の周期のその組み合わせの日付にポイントを移動します。*g m n c*を使用すると、次の周期の s ぽの組み合わせの日付にポイントを移動します。あり得ない haab 日付と tzolkin 日付の組み合わせをタイプした場合は、エラーをシグナルします。

Emacs は Maya 暦の名前の入力を求めるときは強い補完 (Section “Completion Exit” in *the Emacs Manual* を参照してください) を使うので、スペルについて心配する必要はありません。

日付の表示フォーマット

`calendar-date-display-form`をセットすることにより、ダイアリー、モードライン、メッセージに表示される日付をカスタマイズできます。この変数は、文字列形式の数字をもつ変数`month`、`day`、`year`と、アルファベット文字列をもつ`monthname`、`dayname`を含む式のリストを保持します。アメリカ様式では、このリストのデフォルト値は以下のようになります:

```
((if dayname (concat dayname ", ")) monthname " " day ", " year)
```

ヨーロッパ様式では、この値のデフォルトは以下のようになります:

```
((if dayname (concat dayname ", ")) day " " monthname " " year)
```

デフォルトの ISO 日付は以下のようになります:

```
((format "%s-%.2d-%.2d" year (string-to-number month)
         (string-to-number day)))
```

他の典型的なアメリカ様式は以下のものです:

```
(month "/" day "/" (substring year -2))
```

時刻の表示フォーマット

カレンダーとダイアリーは、デフォルトで 1 日の時刻を、‘am’または‘pm’と、1 から 12 の時刻と分によるアメリカ様式の慣習にしたがって表示します。00 から 23 の時刻による、ヨーロッパ様式 (US 軍用様式とも呼ばれる) にしたい場合は、変数 calendar-time-display-formを変更することができます。この変数は式のリストです。このリストには文字列形式の数字をもつ変数 12-hours、24-hours、minutesと、アルファベット文字列をもつ am-pm、time-zoneを含めることができます。デフォルト値は以下のとおりです:

```
(12-hours ":" minutes am-pm
          (if time-zone " (") time-zone (if time-zone ")"))
```

以下の値はヨーロッパ形式の時刻を提供します:

```
(24-hours ":" minutes
          (if time-zone " (") time-zone (if time-zone ")"))
```

1 日の時刻を返すカレンダー関数は少ないことに注意してください (現在のところ solar 関数のみ)。

ダイアリーのカスタマイズ

ダイアリーウィンドウは通常、ダイアリーエントリーの日付が休日に相当する場合は、モードラインとバッファー自身にそれを示します。休日をチェックするプロセスは時間がかかることがあり、それは定義された休日に依存します。このような場合、diary-show-holidays-flagを nilにセットすることにより、ダイアリーの表示を速くすることができます。

変数 diary-number-of-entriesは、1 度に表示されるダイアリーエントリーの日数を制御します。これは calendar-view-diary-initially-flagが tのときの初期表示と、コマンド *M-x diary*に影響します。たとえば値 1(デフォルト) は、その日のダイアリーエントリーだけを表示し、値 2 は翌日のエントリーも表示します。値には 7 つの整数の vector も指定できます。たとえば値が [0 2 2 2 2 4 1]の場合、日曜日にはダイアリーエントリーは表示されず、月曜日から木曜日までは当日と翌日のダイアリーエントリーが表示され、金曜日には金曜日から月曜日のエントリーが表示され、土曜日にはその日のエントリーだけが表示されます。

変数 diary-date-formsをセットすることにより、ダイアリーファイルの日付形式をカスタマイズできます。この変数は日付を認識するパターンのリストです。各日付パターンは、要素が正規表現 (Section “Regular Expressions” in *the Emacs Lisp Reference Manual* を参照してください)、またはシンボル month、day、year、monthname、daynameのリストです。これらすべての要素は、ダイアリーファイルの特定の種類のテキストにマッチするパターンに供されます。日付パターン全体がマッチするためには、リストの各要素が連続してマッチしなければなりません。

日付パターンの正規表現は、標準の構文テーブルを変更してそれ自身の通常の方法でマッチするので、‘*’は単語の構成要素になります。

シンボル month、day、year、monthname、daynameは、日付と考えられる月番号、日付番号、年番号、月の名前、曜日名にマッチします。数字にマッチするシンボルは、0 で開始することもできます。名前にマッチするものは、大文字名と、(calendar-month-abbrev-arrayと calendar-day-abbrev-arrayで指定されるような) 省略形を許容します。ダイアリーエントリーの‘*’は “任意の月” の ‘任意の日付” などを意味するので、すべてのシンボルは‘*’にマッチすることができ、日付とみなされないものにもマッチするべきです。

アメリカ様式での diary-date-formsのデフォルト値は、diary-american-date-formsにより提供されます:

```
((month "/" day "[^/0-9]")
 (month "/" day "/" year "[^0-9]")
```

```
(monthname " *" day "[^,0-9]")
(monthname " *" day ", *" year "[^0-9]")
(dayname "\\W"))
```

変数diary-european-date-formsおよびdiary-iso-date-formsは、他のデフォルト様式を提供します。

このリストの日付パターンは、相互排他的 *(mutually exclusive)* でなければならず、ダイアリーエントリーの任意の部分にマッチせず、日付と 1 文字の空白文字だけにマッチしなければなりません。相互排他的であるならば、パターンは日付を終端する空白文字を越えて、ダイアリーエントリーの一部にマッチしなければならず、日付パターンの最初の要素はバックアップ (backup) されなければなりません。これにより日付識別処理はマッチお終えた後、ダイアリーエントリーの先頭の単語をバックアップします。backupを使う場合でも、日付パターンはダイアリーエントリーの最初の単語の部分を越えてマッチしてはなりません。たとえばdiary-european-date-formsは以下のようなデフォルト値です:

```
((day "/" month "[^/0-9]")
 (day "/" month "/" year "[^0-9]")
 (backup day " *" monthname "\\W+\\<\\([^*0-9]\\|\\([0-9]+[:aApP]\\)\\)")
 (day " *" monthname " *" year "[^0-9]")
 (dayname "\\W"))
```

3 番目のパターンでの backupの使用に注目してください。これは 4 番目のパターンと区別するために、日付自体を越えて単語の一部にマッチする必要があるからです。

非グレゴリオ暦を使用するカレンダーエントリー

グレゴリオ暦にもとづくエントリーと同様に、ダイアリーは Bah^^c3^^a1'^^c3^^ad、Hebrew、Islamic の日付にもとづくエントリーをもつことができます。そのようなエントリーの識別は時間がかかる場合がありますが、これらを使用する人はほとんどいないので、使用する場合は明示的に有効にしなければなりません。ダイアリーが、ダイアリーエントリーの Hebrew 日付を識別するようにしたい場合、たとえば以下のようにして、これを行なわなければなりません:

```
(add-hook 'diary-nongregorian-listing-hook 'diary-hebrew-list-entries)
(add-hook 'diary-nongregorian-marking-hook 'diary-hebrew-mark-entries)
```

同様に Islamic と Bah^^c3^^a1'^^c3^^ad のエントリーにたいしては、diary-islamic-list-entriesと diary-islamic-mark-entries、または diary-bahai-list-entriesとdiary-bahai-mark-entriesを追加します。

これらのダイアリーエントリーは、グレゴリオ日付のダイアリーエントリーと同じフォーマットをもちます。例外は、Bah^^c3^^a1'^^c3^^ad 日付の前にはdiary-bahai-entry-symbol(デフォルトは'B')、Hebrew 日付の前にはdiary-hebrew-entry-symbol(デフォルトは'H')、Islamic 日付の前にはdiary-islamic-entry-symbol(デフォルトは'I') が前置されなければならないことです。さらに、非グレゴリオの月の名前は省略できません (最初の 3 文字では一意にならないものがあるからです。一般的な Hebrew 年の Adar を使いたい場合は、"Adar I" を使わなければならないことにも注意してください。) たとえば Hebrew 日付の Heshvan 25 にたいするダイアリーエントリーは、以下のようになります:

```
HHeshvan 25 Happy Hebrew birthday!
```

これはヘブライ暦の Heshvan 25 に対応する日付のダイアリーに表示されます。以下は Islamic 日付の Dhu al-Qada 25 にマッチするダイアリーエントリーです:

```
IDhu al-Qada 25 Happy Islamic birthday!
```

グレゴリオ日付のダイアリーエントリーのように、非グレゴリオのエントリーも前に `diary-nonmarking-symbol`(デフォルトは '`&`') が前置されている場合はマークされません。

以下は、選択された日付または他の日付にたいして、Bah^^c3^^a1'^^c3^^ad、Hebrew、Islamic の日付によるダイアリーエントリーを作成するコマンドの表です:

i h d	`diary-hebrew-insert-entry`
i h m	`diary-hebrew-insert-monthly-entry`
i h y	`diary-hebrew-insert-yearly-entry`
i i d	`diary-islamic-insert-entry`
i i m	`diary-islamic-insert-monthly-entry`
i i y	`diary-islamic-insert-yearly-entry`
i B d	`diary-bahai-insert-entry`
i B m	`diary-bahai-insert-monthly-entry`
i B y	`diary-bahai-insert-yearly-entry`

これらのコマンドは、通常のダイアリーエントリーの対応するコマンドに似ています。これらはカレンダーウィンドウのポイント位置の日付に適用され、これらのコマンドが行なうのはダイアリーファイルの最後にダイアリーエントリーの日付部分を挿入することだけです。その後、ダイアリーエントリーの残りを挿入しなければなりません。特定の非グレゴリオ日付のエントリーを追加する基本的なコマンドとして、与えられた非グレゴリオ日を各月に追加する '`monthly`'コマンド、与えられた非グレゴリオの月日を各年に追加する '`yearly`'コマンドがあります。

ダイアリーの表示

ダイアリーの表示は、ダイアリーエントリーのリストを準備して、変数 `diary-display-function` で指定された関数を実行することにより機能します。デフォルト値の `diary-fancy-display`は、ダイアリーエントリーを表示するためだけに存在する特別なバッファーにコピーして、ダイアリーエントリーと休日を表示します。ダイアリーエントリーを別のバッファーにコピーすることにより、表示されるテキストの見栄えをよくする機会— たとえばダイアリーエントリーに適用される日付順にソートするなど — が提供されます。

通常、fancy diary(装飾的なダイアリー) バッファーは、たとえその日が休日であっても、ダイアリーエントリーがない日は表示しません。そのような日を fancy diary バッファーに表示したいときは、変数 `diary-list-include-blanks`を `t`にセットします。

fancy diary バッファーは View モードを有効にします (Section "View Mode" in *the Emacs Manual* を参照してください)。

他の表示方法の `diary-simple-display`は、実際のダイアリーバッファーを表示して、適合しないエントリーを隠すために非表示のテキストを使います。休日はモードラインに表示されます。この方法の有利な点は、ダイアリーファイルを直接編集して、変更を保存できることです。しかし、この方法は fancy 方式のように柔軟ではありません。たとえば、これはエントリーのソートはできません。他の不利な点としては、非表示のテキストが混乱の元となることがある点です。たとえば、他の場所に張り付けるためにリージョンのテキストをコピーした場合、非表示のテキストも含まれます。同様に、目に見えるダイアリーバッファーは一種の幻影なので、単にバッファーを印刷しても、スクリーンで表示されているものが印刷されるわけではありません。

この理由により、ダイアリーバッファーのハードコピーを見た通りに印刷する特別なコマンド *M-x diary-print-entries*があります。これは、どちらの表示方式でも機能します (たとえ fancy display バッファーが他のバッファーと同じようにプリントできるとしても)。週の各曜日のハードコピーを印刷するには、ポイントを週の初めの曜日に移動して *7 d*とタイプし、それから *M-x diary-print-entries* を実行します。通常のように、休日が含まれる場合、表示は若干遅くなります。変数 `diary-show-holidays-flag`を `nil`にセットすることにより、速度を改善することができます。

このコマンドは、ダイアリーバッファーで現在可視なダイアリーエントリーだけを含んだ一時的なバッファーを作成します。単なる表示とは異なり、他の無関係なエントリーは隠されるだけではなく、まったく含まれません。バッファーを作成したら、フック `diary-print-entries-hook`が実行されます。このフックのデフォルト値はコマンド `lpr-buffer`で、これはデータを直接プリンターに送信します (Section "Printing" in *the Emacs Manual* を参照してください)。違うコマンドを印刷に使いたい場合は、単にこのフックの値を変更するだけです。他のコマンドには、たとえば行を日付と時刻順に再配置することなどが含まれるかもしれません。

simple ダイアリーウィンドウで表示されているときと同じように、ダイアリーエントリーを編集できますが、表示されているバッファーには、表示から隠されている部分が含まれていることを覚えておくのは重要です。これは、たとえば *C-f* (`forward-char`) は表示されている最後の行にポイントを移動できますが、実際は隠されている行の途中かもしれないことを意味します。

simple 表示でダイアリーエントリーを編集するときは注意してください。表示されている行の途中での行の追加、文字の追加や削除は問題になりませんが、行の最後で編集する場合は、意図した通りにはならないでしょう。行の削除は、それに続く他の非表示のエントリーを削除するかもしれません。simple diary バッファーで編集する前に、*s* (`diary-show-all-entries`) でファイル全体を表示するのが最善です。

Fancy Diary 表示

以下の機能は、fancy diary 表示だけで機能します。

ノーマルフック `diary-list-entries-hook`を使用して、各曜日のダイアリーエントリーを日時順でソートできます。以下はその方法です:

```
(add-hook 'diary-list-entries-hook 'diary-sort-entries t)
```

これは各曜日にたいして、認識可能は日時で始まるダイアリーエントリーをソートします。時刻がないダイアリーエントリーは、各曜日の最初に配置されます。ソートコマンドがフックリストの最後に配置されているわけに注意してください。リストの最初の方に配置されている場合、これはダイアリーエントリーの順序を変更したり、アイテムを追加します。

コメント区切りとなる文字列を `diary-comment-start`と `diary-comment-end`にセットすることにより、ダイアリーエントリーに 'コメント'を記述できます。fancy 表示はコメントを印刷しません。他のパッケージ (たとえば appointment パッケージ。Section "Appointments" in *the Emacs Manual* を参照してください) で使用するメタデータをコメント内に記述したいと思うかもしれません。

メインとなるダイアリーファイルに、他のファイルをインクルードできます。これにより、グループのメンバー全員に適用されるイベント用のダイアリーファイルを、共有することができます。ダイアリーファイルで、`diary-include-string`で開始される行:

```
#include "filename"
```

は、ファイル *filename*からダイアリーエントリーを fancy diary バッファーにインクルードします。インクルードの仕組みは再帰的なので、インクルードされたファイルは他のファイルをインクルードできます (もちろん循環的なインクルードについては注意しなければなりません)。インクルード機能を有効にするには、以下のようにします:

```
(add-hook 'diary-list-entries-hook 'diary-include-other-diary-files)
(add-hook 'diary-mark-entries-hook 'diary-mark-included-diary-files)
```

インクルード機能は fancy diary だけで機能します。なぜなら simple diary 表示は、エントリーをダイアリーファイルから直接表示するからです。

sexp エントリーと Fancy Diary 表示

sexp(s-expression: S 式) ダイアリーエントリーにより、どのダイアリーエントリーに適用するか、複雑な条件判定以上のことができます。sexp エントリーは、ダイアリーファイルで `diary-sexp-entry-symbol`(デフォルトは '`%%`') が前置されている行です。fancy diary 表示では、sexp エントリーはエントリーの日付に応じて、エントリーのテキストを生成できます。

たとえば記念日のダイアリーエントリーでは、記念日から経過した年数をダイアリーエントリーのテキストに挿入できます。したがって、以下のダイアリーエントリーの '`%d`':

```
%%(diary-anniversary 10 31 1948) Arthur's birthday (%d years old)
```

は年齢に置換されるので、1990 年 10 月 31 日の fancy diary バッファーでは、以下のように表示されます:

```
Arthur's birthday (42 years old)
```

かわりに、ダイアリーファイルに以下のようなエントリーが含まれている場合:

```
%%(diary-anniversary 10 31 1948) Arthur's %d%s birthday
```

は 1990 年 10 月 31 日の fancy diary バッファーで、以下のように表示されます:

```
Arthur's 42nd birthday
```

同様に、周期的なダイアリーエントリーは、それが繰り返し発生した回数を挿入できます:

```
%%(diary-cyclic 50 1 1 2012) Renew medication (%d%s time)
```

これは以下のように表示されます:

```
Renew medication (5th time)
```

これは 2012 年 9 月 7 日の fancy diary 表示です。

発生する日付だけでなく、それより前のダイアリーエントリーを含めるための、"事前リマインダー" となる sexp ダイアリーエントリーもあります。たとえば記念日の 1 週間前にリマインダーが欲しいときは、以下を使用します

```
%%(diary-remind '(diary-anniversary 12 22 1968) 7) Ed's anniversary
```

これにより、fancy diary は 12 月 15 日と 12 月 22 日に、'`Ed's anniversary`'を表示します。

関数 `diary-date`は、整数または `t`(すべての値を意味します) からなる month、day、year の組み合わせで日付を指定します。たとえば、

```
%%(diary-date '(10 11 12) 22 t) Rake leaves
```

これにより fancy diary は

```
Rake leaves
```

を毎年 10 月 22 日、11 月 22 日、12 月 22 日に表示します。

関数 `diary-float`を使って、11 月の第 3 金曜日や、4 月の最後の火曜日といった日付をダイアリーエントリーに記述することができます。パラメーターは *month*、*dayname*、およびインデックス *n*です。エントリーは *month*の最初の日の後の *n*番目の曜日 *dayname*に表示されます。ここで *dayname*=0 は日曜日、1 は月曜日、. . . です。*n*が負の場合、*month*の最後から後方に数えます。*month*の値に指定できるのは、月のリスト、単一の月、`t`の場合はすべての月を意味します。オプションのパラメーター *day*を使用して、*month*の *day*日目の後または前の、*n*番目の曜日 *dayname*を指定することもできます。*day*のデフォルト値は、*n*が正のときは 1 で、*n*が負のときはと気はの最後の日です。たとえば、

```
%%(diary-float t 1 -1) Pay rent
```

これにより fancy diary は

```
Pay rent
```

を毎月最後の月曜日に表示します。

sexp ダイアリーエントリーの一般性により、アルゴリズム的に記述したダイアリーエントリーを指定できます。sexp ダイアリーエントリーには、任意の与えられた日付にたいして、エントリーを適用するかどうかを計算する式が含まれます。値が非 `nil`の場合はその日付にエントリーが適用され、そうでない場合は適用されません。式では判定する日付を変数 `date`で使用することができます。この変数の値は、グレゴリオ暦を参照するリスト (*month day year*) です。

sexp ダイアリーエントリーは、式の値が非 `nil`のときはその日付に適用されますが、いくつかの値は特別な意味をもちます。値が文字列の場合、その文字列はその日に発生するイベントを説明する文字列です。値は (*mark . string*) という形式をもつこともできます。*mark*は、カレンダーでその日をどのようにマークするかを指定し、*string*はそのイベントの説明です。*mark*が 1 文字の文字列の場合、その文字はカレンダーの日付の隣に表示されます。*mark*がフェイス名の場合、その日はそのフェイスで表示されます。*mark*が `nil`の場合、その日を特にハイライト表示しません。

21 日がウィークデイのときは 21 日、21 日が週末のときは前の日の金曜日に給料が支払われるとしましょう。以下はそのような日付にマッチする sexp ダイアリーエントリーです:

```
&%%(let ((dayname (calendar-day-of-week date))
         (day (cadr date)))
       (or (and (= day 21) (memq dayname '(1 2 3 4 5)))
           (and (memq day '(19 20)) (= dayname 5)))
       ) Pay check deposited
```

以下の sexp ダイアリーエントリーは、(fancy diary 表示において) 日付により異なるテキストをもつダイアリーエントリーを作成することができます:

`%%(diary-sunrise-sunset)`
: 地方時で、今日の日の出と日の入りの時刻のダイアリーエントリーを作成します。

`%%(diary-lunar-phases)`
: 月の位相にたいするダイアリーエントリーを作成します。

`%%(diary-day-of-year)`
: その年での通算日数と、その年の残り日数でダイアリーエントリーを作成します。

`%%(diary-iso-date)`
: 今日と等価な、ISO 商用日付のダイアリーエントリーを作成します。

`%%(diary-julian-date)`
: 今日と等価な、ユリウス暦日のダイアリーエントリーを作成します。

`%%(diary-astro-day-number)`
: 今日と等価な、天文日 (ユリウス日) のダイアリーエントリーを作成します。

`%%(diary-bahai-date)`
: 今日と等価な、Bah^^c3^^a1'^^c3^^ad 暦日のダイアリーエントリーを作成します。

`%%(diary-chinese-date)`
: 今日と等価な、旧暦日のダイアリーエントリーを作成します。

`%%(diary-coptic-date)`
: 今日と等価な、Coptic カレンダー日のダイアリーエントリーを作成します。

`%%(diary-ethiopic-date)`
今日と等価な、エチオピア暦日のダイアリーエントリーを作成します。

`%%(diary-french-date)`
今日と等価な、フランス革命暦の日付のダイアリーエントリーを作成します。

`%%(diary-hebrew-date)`
今日と等価な、ヘブライ暦の日付のダイアリーエントリーを作成します。

`%%(diary-islamic-date)`
今日と等価な、イスラム暦の非助のダイアリーエントリーを作成します。

`%%(diary-mayan-date)`
今日と等価な、マヤ暦の日付のダイアリーエントリーを作成します。

`%%(diary-persian-date)`
今日と等価な、Persian calendar の日付のダイアリーエントリーを作成します。

例えば、以下のようなダイアリーエントリーを含めると

```
&%%(diary-hebrew-date)
```

fancy diary 表示を使用している場合は、毎日のダイアリー表示に、その日に対応するヘブライ暦の日付が含まれるようになります (simple diary 表示を使用している場合は、任意の日付のダイアリーにリテラル行 '`&%%(diary-hebrew-date)`'が表示されます)。

以下の関数は、特定の標準的なヘブライ sexp ダイアリーエントリーを構築するために使用されます:

`%%(diary-hebrew-rosh-hodesh)`
新しいヘブライ月にたいして、礼拝の発生と告知を告げるダイアリーエントリーを作成します。

`%%(diary-hebrew-parasha)`
毎週のシナゴーグ経典 (synagogue scripture) の読書会を告げる、土曜日のダイアリーエントリーを作成します。

`%%(diary-hebrew-sabbath-candles)`
安息日のキャンドルライトを告げる、地方時のダイアリーエントリーを作成します。

`%%(diary-hebrew-omer)`
適切な場合は、omer を数えるダイアリーエントリーを作成します。

`%%(diary-hebrew-yahrzeit` *month day year*`)` *name*
命日をマークするダイアリーエントリーを作成します。命日の日付はグレゴリオ暦の日付です。ダイアリーエントリーは適切なヘブライ暦の命日、およびその前日に表示されます (カレンダーの日付様式に対応してパラメーターの順序は変化します。たとえばヨーロッパ標識では *day*、*month*、*year*の順です)。

`%%(diary-hebrew-birthday` *month day year*`)`
ヘブライ暦での誕生日のダイアリーエントリーを作成します。

上記でドキュメントされたすべての関数は、オプションの引数 *mark*を受け取ります。これはカレンダー表示で、その日をどのようにマークするかを指定します。上記の関数の 1 つが特定の日付に適用されると決定された場合、上述したように *mark*を含んだ値を戻します。

Emerge でのファイルのマージ

行き違いの指示を受けて、同じプログラムを異なる 2 つの方向に修正してしまうのは、プログラマーにとって珍しいことではありません。この混乱を正常な状態に戻すには、2 つのバージョンをマージする必要があります。Emerge はこれを簡単にします。ファイルを比較する他の方法については、Section “Comparing Files” in *the Emacs Manual*、および Section “Ediff” in *The Ediff Manual* を参照してください。

Emerge の概要

Emerge を開始するには、以下の 4 つのコマンドの 1 つを実行します:

M-x emerge-files
: 指定した 2 つのファイルをマージします。

M-x emerge-files-with-ancestor
: 共通の祖先 (ancestor) を参照して、指定した 2 つのファイルをマージします。

M-x emerge-buffers
: 2 つのバッファーをマージします。

M-x emerge-buffers-with-ancestor
: 第 3 のバッファーにある共通の祖先を参照して、2 つのバッファーをマージします。

Emerge コマンドは 2 つのファイルまたはバッファーを比較して、3 つのバッファーにそれ表示します。最初の 2 つは入力テキスト (*A* バッファーと *B* バッファー) で、残りの 1 つ (マージバッファー) はどこにマージが行われたかを表示します。マージバッファーは相違だけではなく、マージされたテキストをすべて表示します。2 つの入力テキストが異なる場所では、どちらをマージバッファーに含めるか選択できます。

既存のバッファーから入力を得る Emerge コマンドは、そのバッファーがナローされている場合は、バッファーのアクセス可能な部分だけを使用します。Section “Narrowing” in *the Emacs Manual* を参照してください。

2 つのマージされるテキストの元となる、共通の祖先となるバージョンが利用可能な場合は、Emerge はどちらが正しい候補かを推測するために、それを使用することができます。一方のカレントバージョンが祖先に一致する場合、Emerge はもう一方のカレントバージョンが、マージされたバージョンに残すべき、意図した変更であると仮定します。共通の祖先となるテキストを指定したい場合は、‘`with-ancestor`’がつくコマンドを使用します。これらのコマンドは 3 つのファイルまたはバッファーの名前— バージョン A、バージョン B、そして共通の祖先の名前を読み取ります。

比較が終了してバッファーの準備ができた後、対話的なマージが開始されます。マージバッファーで特別なマージコマンドをタイプすることにより、マージを制御できます ([Merge Commands], page 22 を参照してください)。入力テキストの相違それぞれにたいして、どちらを残すか、または両方編集するか選択することができます。

マージバッファーはこれらの選択を行うために、Emerge モードという特別なメジャーモードを使用します。しかし、そのバッファーでは通常の Emacs コマンドで編集することもできます。

常に Emerge の注目は、選択された相違と呼ばれる、特定の相違に焦点を置きます。この相違は、3 つのバッファーで以下のようにマークされます:

```
vvvvvvvvvvvvvvvvvvvv
text that differs
^^^^^^^^^^^^^^^^^^^^
```

Emerge はすべての相違に順番に番号をふり、モードラインには選択された相違の番号が常に表示されます。

マージバッファーは通常、バージョン A のテキストで開始されます。しかしバージョン A の相違が共通の祖先と一致する場合、その相違の初期値にはバージョン B が優先されます。

マージバッファーを抜けるとき、Emerge はマージされたテキストを残します。このとき、*C-x C-w* で、それをファイルに保存できます。`emerge-files`または `emerge-files-with-ancestor`に数引数を与えた場合、ミニバッファーを使用して、出力するファイル名を読み取ります (これは、このコマンドが読み取るファイル名の最後のファイル名です)。その後 Emerge から抜けると、マージされたテキストを出力ファイルに保存します。

通常、Emerge コマンドは終了するとき出力バッファーを、そのバッファーのファイルに保存します。Emerge を *C-]*で中断した場合、Emerge コマンドは出力バッファーを保存しませんが、もし保存したいときは自分で保存することができます。

Emerge のサブモード

マージコマンドに与える 2 つのモード — Fast モードと Edit モードを選択することができます。Fast モードでは、基本的なマージコマンドは 1 文字ですが、通常の Emacs コマンドは利用不可です。これはマージコマンドだけを使いたいときは便利です。Edit モードでは、すべてのマージコマンドはプレフィクスキー *C-c C-c*で始まり、通常の Emacs コマンドも利用可能です。これによりマージバッファーで編集することができますが、Emerge 操作は遅くなります。

*e*で Edit モード、*C-c C-c f*で Fast モードに切り替わります。モードラインには Edit モードおよび Fast モードは、'`E`'と '`F`'で示されます。

Emerge は、特定のマージコマンドがどのように機能するかに影響を与える、2 つの追加的なサブモード — Auto Advance モードと Skip Prefers モードがあります。

Auto Advance モードが効力をもつ場合、`a`または *b*コマンドで次の相違に移動します。これは候補を単に選んでいくだけで、素早くマージすることができます。Auto Advance モードの場合、モードラインに '`A`'が示されます。

Skip Prefers モードが効力をもつ場合、*n*および *p*コマンドは、状態が "prefer-A" と "prefer-B" の相違をスキップします ([State of Difference], page 21 を参照してください)。したがって、どちらのバージョンも "正しい" と推定できない相違だけを見ていくことができます。Skip Prefers モードの場合、モードラインに '`S`'が示されます。このモードは祖先がある場合にだけ適しています。

Auto Advance をセットまたはクリアーするには、コマンド *s a* (`emerge-auto-advance`) を使用します。Skip Prefers モードをセットまたはクリアーするには、*s s* (`emerge-skip-prefers`) を使用します。これらのコマンドは、正の引数の場合はモードをオンにして、負または 0 の引数のときはモードをオフにし、引数がないときはモードのオン・オフを切り替えます。

相違の状態

マージバッファーでは、相違は文字 '`v`'および '`^`'のラインでマークされます。各相違は、以下の 7 つの状態のうち 1 つをもちます:

A 相違にはバージョン A が表示されています。`a`コマンドは常にこの状態を生成します。モードラインには '`A`'が示されます。

B 相違にはバージョン B が表示されています。*b*コマンドは常にこの状態を生成します。モードラインには '`B`'が示されます。

default-A
default-B 選択を行っていないので、相違にはデフォルトとして A または B の状態が表示されています。すべての相違は default-A 状態で開始されます (したがってマージバッファーは A バッファーのコピーとなります) が、例外として、もう一方のほうが “preferred(好ましい)” 場合を除きます (以下参照)。

相違を選択したときに、状態は default-A または default-B から、A または B に遷移します。したがって、選択された相違が状態 default-A または default-B をもつことはなく、これらの状態がモードラインに表示されることはありません。

コマンド *d a*は、デフォルトの状態に default-A を選択し、*d b*は default-B を選択します。これらで選択されるデフォルトは、まだ 1 度も選択していないか、好ましい候補をもたないすべての相違に適用されます。順番に移動しながらマージを行っている場合、一度も選択されていない相違とは、選択された相違の後のすべての相違です。したがって順番に移動しながら、セクションの間で *d a*と *d b*を使い分けることにより、マージバッファーのあるセクションにたいしてはバージョン A をデフォルトに、他のセクションにはバージョン B を効果的に選択できます。

prefer-A
prefer-B 相違には、状態 A または状態 B が表示されています。なぜなら、それが *preferred(好ましい)* からです。これはまだ明示的な選択を行なっていないが、一方の候補は共通の祖先と同じなので、もう一方の候補の方が正しく見えることを意味します。したがって A バッファーが共通の祖先と同じ場合、バージョン B の方が好ましいといえます。なぜなら、それは実際に変更されているからです。

これら 2 つの状態は、モードラインに ‘A*’と ‘B*’で表示されます。

combined *x c*または *x C*コマンドの結果として、相違は状態 A および状態 B が混合されたものを表示しています。

相違が 1 度この状態になると、*a*コマンドと *b*コマンドは、数引数を与えないかぎり何もしません。

モードラインには状態 ‘comb’が表示されます。

マージコマンド

以下は Fast モードのマージコマンドです。Edit モードでは前に *C-c C-c*をつけてください:

p 前の相違を選択します。

n 次の相違を選択します。

a この相違にバージョン A を選択します。

b この相違にバージョン B を選択します。

C-u n j 番号 *n*の相違を選択します。

. ポイントを含む相違を選択します。

q 終了 — マージを終了します。

C-] 中断 — マージを exit して出力を保存しません。

f Fast モードに移行します (Edit モードでは実際には *C-c C-c f*です)。

e Edit モードに移行します。

`l` 3つのウィンドウにたいして、(*C-l*のように) 再センタリングをします。引数を指定すると、デフォルトの3ウィンドウ表示を再構築します。

`-` 数引数の一部を指定します。

digit これも数引数の一部を指定します。

`d a` ポイント位置からマージバッファーの下方へ、バージョンAをデフォルトとして選択します。

`d b` ポイント位置からマージバッファーの下方へ、バージョンBをデフォルトとして選択します。

`c a` この相違のバージョンAを、killリングにコピーします。

`c b` この相違のバージョンBを、killリングにコピーします。

`i a` この相違のバージョンAを、ポイント位置に挿入します。

`i b` この相違のバージョンBを、ポイント位置に挿入します。

`m` 相違の周囲に、ポイントとマークを配します。

`^` 3つのウィンドウすべてを、(*M-v*のように) 下にスクロールします。

`v` 3つのウィンドウすべてを、(*C-v*のように) 上にスクロールします。

`<` 3つのウィンドウすべてを、(*C-x <*のように) 左にスクロールします。

`>` 3つのウィンドウすべてを、(*C-x >*のように) 右にスクロールします。

`|` 3つのウィンドウすべての水平スクロールをリセットします。

`x 1` マージウィンドウを1行に縮めます (フルサイズに復元するには *C-u l*を使用します)。

`x c` この相違の2つのバージョンを合成します ([Combining in Emerge], page 24 を参照してください)。

`x f` Emergeが操作しているファイルまたはバッファーの名前を、Helpウィンドウに表示します (ウィンドウを復元するには *C-u l*を使用します)。

`x j` この相違を次の相違に結合します (*C-u x j*は前の相違に結合します)。

`x s` この相違を2つの相違に分割します。このコマンドを使う前に、3つのバッファーで、相違を分割したい位置にポイントを配してください。

`x t` 相違の上部と下部にある、同じ行を取り除きます。このような行は、バージョンAとバージョンBが同じでも、それらが祖先となるバージョンと異なる場合に発生します。

Emergeの終了

`q`コマンド (`emerge-quit`) はマージを終了して、指定されている場合は結果を出力ファイルに保存します。これはバッファーAおよびBの内容を正しいものに復元するか、もしそれらがEmergeにより作成され、何も変更されていないときはkillします。これによりマージバッファーでEmergeコマンドは使えなくなります。なぜならそれらを実行することにより、さまざまなバッファーの内容にダメージを与えるからです。

*C-]*はマージを中断します。これは出力ファイルに書き込まずに終了することを意味します。出力ファイルを指定していない場合、マージの終了と中断に実際の違いはありません。

Emergeコマンドが他のLispプログラムから呼び出された場合、正常終了の戻り値は`t`で、中断 (abort) のときは`nil`です。

2 つのバージョンの合成

特定の相違にたいして両方の候補を使いたいときがあります。これを行なうにはマージバッファーを以下のように編集する *x c*を使用します:

```
#ifdef NEW
version from B buffer
#else /* not NEW */
version from A buffer
#endif /* not NEW */
```

この例は C プリプロセッサー条件が 2 つのバージョン候補を区切っていますが、区切りに使う文字列は、変数 `emerge-combine-versions-template`に選択した文字列をセットすることにより、指定することができます。この文字列で '`%a`'はバージョン A、'`%b`'はバージョン B です。上記の結果を生成するデフォルトのセッティングは、以下のようなものです:

```
"#ifdef NEW\n%b#else /* not NEW */\n%a#endif /* not NEW */\n"
```

Emerge の細かい注意点

マージの間、A および B バッファーを自分で編集してはいけません。Emerge は一時的にこれらを変更しますが、最終的には元の内容に戻します。

複数のマージを一度に行うことができます — ただし、ある 1 つのバッファーを複数のマージの入力として使用しないでください。なぜなら、これらのバッファーに加えられる一時的な変更が、1 つのバッファーに加えられてしまうからです。

ファイル全体を比較する必要があるため、Emerge の開始には時間がかかることもあります。Emacs は、`diff`が終了するまで他のことを行なえません。入力ファイルが大きいときはバックグラウンドで比較を行なうように、誰かが Emerge を変更するかもしれません— そうなれば Emerge がコマンドを受け付けられるようになるまでの間、Emacs で他の作業を続けることができるでしょう。

マージをセットアップした後、Emerge はフック `emerge-startup-hook`を実行します。Section "Hooks" in *the Emacs Manual* を参照してください。

VC の上級の使用法

一般的に使用される Emacs のバージョンコントロール (VC) サポートの仕様については、メインの Emacs マニュアル (Section “Version Control” in *the Emacs Manual* を参照してください) に記述されています。このチャプターでは、より上級の VC の使用法について説明します。

VC のその他のコマンドと機能

このセクションでは、それほど頻繁には使用されない VC の機能を説明します。

変更ログと VC

`ChangeLog`ファイル (Section “Change Log” in *the Emacs Manual* を参照してください) をもつプログラムにたいして RCS や CVS を使用する場合、バージョンコントロールの以前のコミットのログエントリーから、ChangeLog のエントリーを生成することができます。

これは RCS と CVS だけで機能することに注意してください。この手段は特に、現代的な変更セットベースのバージョンコントロールシステムには不適切です。なぜなら、`ChangeLog`ファイルへの変更自体が、通常は変更セットの一部としてコミットされるからです。この場合、最初に ChangeLog エントリーを記述して、コミットするときにそれを‘`*vc-log*`’バッファーに引用します (Section “Log Buffer” in *the Emacs Manual* を参照してください)。

C-x v a
: カレントディレクトリーの `ChangeLog`ファイルを visit して、そのディレクトリーの登録されたファイルにたいして、最新の ChangeLog エントリー以降にコミットされたバージョンにたいする、新しいエントリーを作成します。(`vc-update-change-log`)。

C-u C-x v a
: 上記と同様ですが、カレントバッファーのファイルにたいするエントリーだけを探します。

たとえば、`ChangeLog`の最初の行の日付が 1999-04-10 で、それ以降のチェックインは 1999-05-22 の Nathaniel Bowditch による `rcs2log`だけで、それのログエントリーが‘`Ignore log messages that start with ‘#’.`’だったとします。その場合、*C-x v a*は `ChangeLog`エントリーとして、以下を挿入します:

```
1999-05-22  Nathaniel Bowditch  <nat@apn.org>

        * rcs2log: Ignore log messages that start with ‘#’.
```

バージョンコントロールのログエントリーが、(行頭にカッコで括られて記述される) 関数名を指定する場合、それは `ChangeLog`のエントリーに反映されます。たとえば `vc.el`にたいするログエントリーが‘`(vc-do-command): Check call-process status.`’の場合、`ChangeLog`のエントリーは以下のようになります:

```
1999-05-06  Nathaniel Bowditch  <nat@apn.org>

        * vc.el (vc-do-command): Check call-process status.
```

*C-x v a*が複数の ChangeLog エントリーを 1 度に追加するとき、それらがすべて同じ作者に、ほぼ同じ日時にチェックインされている場合、関連するログエントリーとしてそれらをグループ化します。複数のファイルにたいするログエントリーがすべての同じテキストの場合、それらを 1 つのエントリーにまとめます。

バージョンコントロールされたファイルの削除とリネーム

M-x vc-delete-file
: ファイル名の入力を求め、そのファイルを作業ツリーから削除して、コミット時に削除するようスケジュールします。

M-x vc-rename-file
: 2 つのファイル名 *var*および *old*の入力を求め、作業ツリーのファイルをリネームし、コミット時にリネームを行うようスケジュールします。

バージョンコントロールされたファイルを削除したい場合、コマンド *M-x vc-delete-file*を使用します。これはファイル名の入力を求め、バージョンコントロールシステムを通じてそれを削除します。ファイルは作業ツリーから削除され、VC Directory バッファー (Section “VC Directory Mode” in *the Emacs Manual* を参照してください) では、状態に ‘`removed`’が表示されます。それをコミットするとき、リポジトリー内で削除が実行されます。

バージョンコントロールされたファイルをリネームするには、*M-x vc-rename-file*とタイプします。これは 2 つの引数の入力を求めます。それはリネームしたいファイルの名前を、それの新しい名前です。そしてバージョンコントロールシステムを通じてリネームを処理します。作業ツリーではリネームは即座に行われます。リネームされたファイルをコミットするとき、リポジトリーに反映されます。

現代的なバージョンコントロールシステムは、リネームにたいするビルトインのサポートがあり、リネームされたファイルは元のファイルのすべての変更履歴を保持します。CVS および古いバージョンコントロールシステムでは、`vc-rename-file`コマンドは、実際には古いファイルを新しい名前にコピーして、それを登録してから古いファイルを削除することにより機能します。この場合、変更履歴は保存されません。

リビジョンタグ

ほとんどのバージョンコントロールシステムは、バージョンコントロールされたツリーの特定のバージョンにたいして、リビジョンタグ *(revision tag)* を適用できます。現代的な変更セットベースのバージョンコントロールシステムでは、リビジョンタグは単に特定のリビジョンにたいするシンボリック名です。CVS のようなファイルベースの古いシステムで、各タグははバージョンコントロールされた一連のファイル全体に追加され、それらを 1 つの単位として処理できるようにします。リビジョンタグは一般的にユーザーに配布されるリリースを識別するのに使用されます。

タグにたいする 2 つの基本的なコマンドがあります。1 つは与えられた名前でタグを作成し、もう 1 つは名前がつけられたタグを取得するコマンドです。

`C-x v s` *name* `RET`
: カレントディレクトリーまたはその配下のディレクトリーの、すべての登録されたファイルの作業リビジョンにたいする、*name*という名前のタグを定義します (`vc-create-tag`)。

`C-x v r` *name* `RET`
: カレントディレクトリーまたはその配下のディレクトリーの、すべての登録されたファイルにたいして、リビジョンのタグが *name*のものを取得します。*name*がブランチ名で、バージョンコントロールシステムがタグからブランチを区別する場合、このコマンドはブランチを切り替えます。(`vc-retrieve-tag`).

  カレントディレクトリーまたは配下のディレクトリーのファイルがロックされている場合、このコマンドは何もせずにエラーを報告します。これは作業中の上書きを避けるためです。

*C-x v =*または *C-x v ~*の引数として、タグまたはブランチ名を与えることができます (Section “Old Revisions” in *the Emacs Manual* を参照してください)。したがって、カレントファイルとタグ付けされたバージョンを比較したり、タグ付けされたバージョン同士を比較することができます。

SCCS では、VC 自身がタグを実装しているので、VC を通じてのみタグを見ることができます。それより新しいほとんどのシステム (CVS、Subversion、bzr、git、hg を含む) は、ネイティブのタグ機能をもっており、利用可能な場合、VC はそれを使用します。これらのタグは、VC を通さなくても見ることができます。

ファイルベースのバージョンコントロールシステムでは、登録されたファイルをリネームするとき、それのマスターもリネームする必要があります。コマンド `vc-rename-file`は、これを自動的に行います (Section “VC Delete/Rename” in *the Emacs Manual* を参照してください)。SCCS を使用している場合、、そのファイルが新しい名前になったことを告げるために、タグの記録も更新しなければなりません (`vc-rename-file`もこれを行います)。記録された名前では、すでに存在しないマスターファイルを参照する古いタグは無効になります。VC はそれを取得しません。RCS および SCCS でのタグの手修正は、このマニュアルの範囲を超えるでしょう。`vc-rename-file`を使用することにより、ファイルを取得できる有効なタグを作成できますが、それですべての問題が解決されるわけではありません。たとえば、プログラムのいくつかのファイルは、他のファイルを名前で参照するかもしれません。少なくとも makefile はリネームしたファイルを参照するでしょう。古いタグを取得した場合、リネームされたファイルは、makefile が期待しない新しい名前で取得されます。そのため、プログラムはうまく機能しないでしょう。

バージョンコントロールヘッダーの挿入

Subversion、CVS、RCS、SCCS では、バージョンヘッダー *(version headers)* と呼ばれる文字列を、ファイル内に置くことができます。そのファイルがコミットされたとき、バージョンコントロールシステムは自動的にリビジョン番号、コミットしたユーザーの名前、その他関連する情報をバージョンヘッダーに挿入します。

VC は通常、バージョンヘッダーの情報を使用しません。例外として、RCS を使用している場合、Emacs は RCS のマスターファイルより信頼できる場合が多いという理由で、ファイルのバージョンを決定するためにバージョンヘッダーを使用します。この方法でバージョンヘッダーを使用するのを禁ずるには、変数 `vc-consult-headers`を `nil`に変更します。

カレントバッファーに適切なヘッダー文字列を挿入するには、*C-x v h* (`vc-insert-headers`) とタイプします。このコマンドはSubversion、CVS、RCS、SCCSだけで機能します。変数 `vc-`*backend*`-header`には、バージョンヘッダーに挿入されるキーワードのリストが含まれます。たとえば CVS は `vc-cvs-header`を使用し、これのデフォルト値は`'("\$Id\$")`です (余分なバックスラッシュは、もし Emacs Lisp ファイルがバージョンコントロールにより保守されるときに、文字列定数がヘッダーと解釈されるのを防ぐためのものです)。*C-x v h*コマンドは。ポイント位置の新しい行にリストのタブで囲まれた各キーワードを挿入し、必要ならばコメント区切りで囲みます。

変数 `vc-static-header-alist`は、バッファー名にもとづき追加する文字列を指定します。これの値は、`(`*regexp* `.` *format*`)`という形式の要素からなるリストです。*regexp*がバッファー名にマッチした場合、バージョンヘッダーの一部として *format*も挿入されます。*format*の中の ‘`%s`’は、そのファイルのバージョンコントロールのタイプに置換されます。

VC のカスタマイズ

変数 `vc-handled-backends`は、どのバージョンコントロールシステムが処理するかを決定します。デフォルト値は `(RCS CVS SVN SCCS Bzr Git Hg Mtn Arch)`で、これには、現在サポートされている、すべてのバージョンコントロールが含まれています。VC にこれらのシステムの 1 つ以上を無視さ

せたい場合、リストからそのシステムの名前を除外します。VC 全体を無効にするには、変数に `nil` をセットしてください。

リストのシステム順序には意味があります。これらのシステムの 1 つ以上に登録されているファイルを visit した場合、デフォルトでは VC は `vc-handled-backends`で最初にくるシステムを使用します。ファイルを最初に登録するときも、この順序が意味をもちます (Section “Registering” in *the Emacs Manual* を参照してください)。

一般的なオプション

Emacs は通常、バージョンコントロールにより保守されるソースファイルのバックアップファイルを保存しません。バージョンコントロールを使用したファイルにもバックアップファイルを作成したいときは、変数 `vc-make-backup-files`に非 `nil`値をセットしてください。

そのファイルがバージョンコントロールされていると知らずに、シンボリックリンクを通じてバージョンコントロールされたファイルを編集すると、予期せぬ結果を招くことがあります。変数 `vc-follow-symlinks`は、バージョンコントロールされたファイルを指すシンボリックリンクを visit しようと試みたときの、Emacs の振る舞いを制御します。値が `ask`(デフォルト) の場合、Emacs は確認を求めます。値が `nil`の場合、Emacs は警告メッセージを表示するだけです。値が `t`の場合、Emacs は自動的にリンクをたどって、かわりに実際のファイルを visit します。

`vc-suppress-confirm`が非 `nil`の場合、*C-x v v*および *C-x v i*は確認を求めずにカレントバッファーを保存し、*C-x v u*も確認を求めず処理を行います。

VC モードは多くの処理を、バージョンコントロールシステムにたいする適切なシェルコマンドを実行することにより行います。`vc-command-messages`が非 `nil`の場合、VC はそれが実行するシェルコマンドを示すメッセージと、コマンドが終了したときの追加のメッセージを表示します。

RCS と SCCS にたいするオプション

デフォルトでは、複数ユーザーの活動を調停するために RCS はロックを使用しますが、最初にファイルをロックしなくても変更をチェックインできる、厳密でないロック *(non-strict locking)* と呼ばれるモードもあります。特定のファイルにたいして厳密でないロックに切り替えるには、‘`rcs -U`’を使用します。詳細については、`rcs`の man-page を参照してください。

RCS ファイルのバージョンコントロール状態を推論するとき、VC は最初にそのファイルの RCS バージョンヘッダー文字列を調べます ([Version Headers], page 27 を参照してください)。ヘッダー文字列がない場合、VC は通常、作業ファイルのパーミッションを調べます。これは速い処理です。ファイルのパーミッションが信頼できない状況もあるかもしれません。そのような場合はマスターファイルが調べられます。これはより高価な処理です。マスターファイルから判るのは、もしそのファイルにたいして何らかのロックがある場合、作業ファイルが実際にロックされたバージョンを含むかどうか、だけです。

`vc-consult-headers`を `nil`にセットすることにより、VC がファイル状態を決定するのにバージョンヘッダーを使用しないように指定できます。その場合、VC は常に、(それが信用できると思われる場合は) ファイルのパーミッションを使うか、マスターファイルをチェックします。

変数 `vc-mistrust-permissions`を設定することにより、ファイルのパーミッションを信頼すべきかの判断基準を指定できます。値が `t`(常にファイルのパーミッションを疑い、マスターファイルをチェックする)、`nil`(常にファイルのパーミッションを信頼する)、または 1 つの引数をとってその判断を行う関数です。引数は `RCS`サブディレクトリーのディレクトリー名です。その関数が非 `nil`値を戻した場合、パーミッションを信頼しません。作業ファイルのパーミッションが誤って変更されたのに気づいた場合、`vc-mistrust-permissions`を `t`にセットします。そうすれば VC はファイル状態を決定するために、常にマスターファイルをチェックします。

VC が SCCS の配下にあるファイルのバージョンコントロール状態を決定する方法は、RCS とほぼ同じです。しかし SCCS のバージョンヘッダーは考慮しません。したがって SCCS を使用する場合、`vc-mistrust-permissions`は効果がありますが、`vc-consult-headers`は効果がありません。

CVS に特有のオプション

変数 `vc-cvs-global-switches`で、すべての CVS 操作に渡す追加のコマンドラインオプションを指定できます。これらのスイッチは `cvs`コマンドの直後、呼び出す操作名の前に挿入されます。

リモートマシンの CVS リポジトリーを使用する場合、VC はネットワークでの通信を最小にしようと試みます。これは変数 `vc-cvs-stay-local`により制御されます。他の変数 `vc-stay-local`もあり、これは CVS を含む、それをサポートする他のバックエンドにもこの機能を有効にします。以下では `vc-cvs-stay-local`についてだけ説明しますが、すべて `vc-stay-local`にも適用できます。

`vc-cvs-stay-local`が `only-file`(デフォルト) の場合、VC はローカルの CVS サブディレクトリーのエントリーと、前の CVS コマンドから戻された情報だけを使って、各ファイルのバージョンコントロール状態を決定します。結果として、あなたがファイルを変更しているとき、他の誰かが他の変更をチェックインした場合、それのコミットを試みるまで衝突を通知されません。

`vc-cvs-stay-local`を `nil`に変更した場合、ローカルのリポジトリーと同じように、`vc-next-action` (*C-x v v*) が何を行うか決定する前に、VC はリモートのリポジトリーに問い合わせを行います。

`vc-cvs-stay-local`に、リポジトリーのあるホスト名にマッチする正規表現を指定することもできます。この場合、ホスト名がパターンにマッチしたときは、VC はローカルに留まります。

リモートのリポジトリーを使用する場合、Emacs は通常、編集された各ファイルのオリジナルバージョンである、自動バージョンバックアップ *(automatic version backups)* を作成します。これらのローカルのバックアップは、変更を最初にファイルに保存したときに作成され、リポジトリーに変更をコミットした後で削除されます (これらは通常の Emacs のバックアップファイルとは異なることに注意してください。Section “Backup” in *the Emacs Manual* を参照してください)。*C-x v =*や *C-x v u*のようなコマンドは、ネットワークへのアクセスを避けるため、可能な場合は自動バージョンバックアップを使用します。

`vc-cvs-stay-local`を `nil`にセットすることにより、自動バージョンバックアップの作成を無効にできます。

自動バージョンバックアップは、*file.~version.~*という形式の名前をもちます。これは *C-x v ~*が古いバージョンを保存するときの名前と似ています (Section “Old Revisions” in *the Emacs Manual* を参照してください)。例外は、バージョンの後ろにある追加のドット (‘.’) です。関連する VC コマンドは、これら両方の種類のバージョンバックアップを使用できます。主な違いは、*C-x v ~* により “手動” で作成されたバージョンバックアップは、コミットしたとき自動的に削除されないことです。

デフォルトで CVS はロックを使用しませんが、`CVSREAD`または *watch* の機能を使用して、ロックのような振る舞いを有効にする方法があります。詳細については、CVS のドキュメントを参照してください。そのような場合、ロックベースのバージョンコントロールシステムで行うように、Emacs で *C-x v v*を使用して、ロックを切り替えることができます (Section “VC With A Locking VCS” in *the Emacs Manual* を参照してください)。

Fortran モード

Fortran モードは、“固定形式 (fixed form)”(または “タブ形式 (tab format)”) のソースコードを編集するためのモードです (通常は Fortran 77)。よりモダンな “自由形式 (free form)” のソースコードを編集するためには、F90 モード (`f90-mode`) を使用します。Emacs は通常、拡張子が ‘`.f`’、‘`.F`’、‘`.for`’のファイルにたいしては Fortran モードを使用し、拡張子が ‘`.f90`’、‘`.f95`’、‘`.f03`’、‘`.f08`’のファイルにたいしては F90 モードを使用します。`auto-mode-alist`をカスタマイズして、拡張子を追加することができます。GNU Fortran は、これら自由形式と固定形式の両方をサポートします。このマニュアルでは主に Fortran モードを記述しますが、対応する F90 モードの機能については、その都度言及します。

Fortran モードは、Fortran 命令文およびサブプログラムにたいする特別な移動コマンドと、Fortran のネスト規則、行番号、行継続された命令文を理解する、インデントコマンドを提供します。Fortran モードは、長い行を適正な Fortran の継続行にブレークする、Auto Fill モードをサポートします。Fortran モードは Hideshow マイナーモード (Section “Hideshow” in *the Emacs Manual* を参照してください)、、および Imenu (Section “Imenu” in *the Emacs Manual* を参照してください) もサポートします。

Fortran のコメントは他の言語とは異なるので、コメントのための特別なコマンドも提供されています。ビルトインの abbrev(省略形) は、Fortran キーワードをタイプする手間を削減します。

*M-x fortran-mode*を使用して、このメジャーモードに切り替えます。このコマンドはフック`fortran-mode-hook`を実行します。Section “Hooks” in *the Emacs Manual* を参照してください。

移動コマンド

“defun”(Fortran のサブプログラム — 関数、サブルーチン、同様に F90 モードのモジュールには、コマンド `fortran-end-of-subprogram`および `fortran-beginning-of-subprogram`を使用します) を単位に移動、操作する通常コマンドに加えて、Fortran モードは命令文や他のプログラム単位に移動する、特別なコマンドを提供します。

C-c C-n 次の命令文の先頭に移動します (`fortran-next-statement`/`f90-next-statement`)。

C-c C-p 前の命令文の先頭に移動します (`fortran-previous-statement`/`f90-previous-statement`)。前の命令文が存在しない場合 (たとえばバッファーの最初の命令文で呼び出された場合)、バッファーの先頭に移動します。

C-c C-e 次のコードブロックの先頭、またはカレントのコードブロックの最後に移動します (`f90-next-block`)。コードブロックとは、サブルーチン、`if`–`endif`命令文などです。これは F90 モードだけのコマンドで、Fortran モードにはありません。数引数を指定すると、複数ブロックを前方に移動します。

C-c C-a 前のブロックに、後方にポイントを移動します (`f90-previous-block`)。これは`f90-next-block`と似ていますが、後方に移動します。

C-M-n カレントのコードブロックの最後にポイントを移動します (`fortran-end-of-block`/`f90-end-of-block`)。数引数を指定した場合、指定した数のブロックを前方に移動します。ポイントを移動する前にマークがセットされます。このコマンドの F90 モードのバージョンでは、ブロックタイプと、(もしあれば) ラベルの整合性をチェックしますが、最外のブロックは不完全かもしれないのでチェックしません。

C-M-p カレントコードブロックの先頭にポイントを移動します (`fortran-beginning-of-block`/`f90-beginning-of-block`)。これは`fortran-end-of-block`と似ていますが、後方に移動します。

コマンド `fortran-beginning-of-subprogram`および `fortran-end-of-subprogram`は、カレントサブプログラムの先頭または後方に移動します。コマンド`fortran-mark-do`および`fortran-mark-if`は、カレントの `do`ブロック、または `if`ブロックの最後にマークをセットして、ポイントをブロックの先頭に移動します。

Fortran のインデント

固定形式 (またはタブ形式) の Fortran コードにたいしては、さまざまな構文エントリー (行番号、行インジケーター、継続行フラグ) が、要求される列に表示されるようにするために、特別なコマンドと機能が必要です。

Fortran のインデントおよびフィルコマンド

C-M-j ポイント位置でカレント行をブレークして、継続行をセットアップします (`fortran-split-line`)。

*M-^* その行を前の行と結合します (`fortran-join-line`)。

C-M-q ポイントのあるサブプログラムの、すべての行をインデントします (`fortran-indent-subprogram`)。

M-q コメントブロックまたは命令文をフィルします (`fortran-fill-paragraph`または`fortran-fill-statement`を使用します)。

キー *C-M-q*は、`fortran-indent-subprogram`を実行します、これはポイントを含む Fortran サブプログラム (関数またはサブルーチン) の、すべての行を再インデントします。

キー *C-M-j*は、`fortran-split-line`を実行します、これは Fortran の流儀にあった方法で行を分割します。非コメント行では、後半は継続行になり、それにしたがったインデントになります。コメント行の場合、両方とも別のコメント行になります。

*M-^*または *C-c C-d*は、コマンド `fortran-join-line`を実行します。これは継続行を前の行に結合します。大雑把にいうと、`fortran-split-line`の逆です。このコマンドを呼び出すとき、ポイントは継続行になければなりません。

Fortran モードでの *M-q*は、ポイントのあるコメントブロックまたは命令文ブロックをフィルします。これは余分な命令文の継続を削除します。

継続行

ほとんどの Fortran77 コンパイラーは、2 つの方法で継続行を記述します。ある行の最初の非スペース文字が列 5 の場合、その行は前の行の継続行です。これを固定形式 *(fixed form)* と呼びます。(GNU Emacs では常に列は 0 から数えますが、Fortran 標準では列 1 から数えることに注意してください)。変数 `fortran-continuation-string`は、列 5 に配す文字を指定します。タブ文字で開始され、その後 '0'以外の任意の数字が記述された行も継続行です。この継続スタイルをタブ形式 *(tab format)* と呼びます (Fortran 90 では、"自由形式 (free form)" という、他の継続行スタイルが導入されました)。

Fortran モードは、どちらの継続行スタイルも使用できます。Fortran モードに入ったとき、バッファー内容から、自動的に適切な継続行スタイルを推論しようと試みます。これはバッファーの開始から、`fortran-analyze-depth`行 (デフォルトは 100) をスキャンすることにより行われます。最初の

行の開始がタブ文字か、6 個のスペースかで選択が決定されます。スキャンが失敗した場合 (たとえば、新しいバッファーで中身が空の場合)、`fortran-tab-mode-default`の値 (`nil`の場合は固定形式で、非 `nil`の場合はタブ形式) が使用されます。モードラインに '`/t`' (`fortran-tab-mode-string`) が表示されている場合、タブ形式が選択されていることを示します。それに応じて Fortran モードは`indent-tabs-mode`の値をセットします。

行のテキストが Fortran の継続マーカー '`$`'で始まるか、列 5 の非空白文字で始まる場合、Fortran モードはそれを継続行として扱います。継続行を `TAB`でインデントした場合、その行をカレントの継続スタイルに変換します。Fortran 命令文を *`C-M-j`*で分割した場合、継続スタイルに応じた継続マーカーがある新しい行が作成されます。

継続スタイルのセッティングは、Fortran モードでの編集の他の側面に影響します。固定形式の場合、命令文の最小列は 6 になります。Fortran ブロック内でそれより大きい列にインデントされる行には、空白文字としてスペース文字だけを使用しなければなりません。タブ形式では、命令文の最小列は 8 で、列 8 より前の空白文字は 1 つのタブ文字でなければなりません。

行番号

その行の最初の非空白文字が数字の場合、Fortran のインデントはそれを行番号と判断して、列 0 から列 4 に移動します (Emacs では列を常に 0 から数えます)。

4 桁以下の行番号は、通常 1 つのスペースでインデントされます。変数 `fortran-line-number-indent`はこれを制御します。これは行番号がもてる最大のインデントを指定します。この変数のデフォルト値は 1 です。Fortran モードは、必要なら指定した最大列以下にインデントを減らして、行番号が列 4 を超えるのを防ごうと試みます。`fortran-line-number-indent`が 5 の場合、行番号は列 4 で終わるように右端に揃えられます。

これらのルールに応じたインデントをするには、単純に行番号を挿入するだけで充分です。各桁が挿入されるたびに、インデントは再計算されます。この機能をオフに切り替えるには、変数 `fortran-electric-line-number`を `nil`にセットしてください。

構文的な慣習

Fortran モードは正しくインデントを行うために、あなたが、Fortran プログラム解読を単純化する特定の慣習にしたがうと仮定します：

- ネストされた 2 つの '`do`'ループは、'`continue`'命令を共有しない。
- '`if`'、'`else`'、'`then`'、'`do`'、その他の Fortran キーワードは、空白文字や行ブレークを含まずに記述される。

  Fortran コンパイラーは一般的に文字列定数の外の空白文字を無視しますが、Fortran モードはこれらのキーワードが隣接していない場合、それらを認識しません。'`else if`'や '`end do`'のような構成は許されますが、2 つ目の単語は継続行ではなく、1 つ目の単語と同じ行にあるべきです。

これらの慣習にしたがわない場合、インデントコマンドは醜いインデントをするかもしれません。しかし正しい Fortran プログラムなら、慣習にしたがわずにインデントされたものでも、その意味は変わりません。

Fortran のインデントのための変数

Fortran のインデントがどのように機能するかを制御する、追加の変数がいくつかあります:

`fortran-do-indent`
: '`do`'命令の各レベルにたいする、追加のインデントです (デフォルトは 3)。

`fortran-if-indent`
: ‘`if`’、‘`select case`’、‘`where`’命令の各レベルにたいする、追加のインデントです (デフォルトは 3)。

`fortran-structure-indent`
: ‘`structure`’、‘`union`’、‘`map`’、‘`interface`’命令の各レベルにたいする、追加のインデントです (デフォルトは 3)。

`fortran-continuation-indent`
: 継続行の本文にたいする、追加のインデントです (デフォルトは 3)。

`fortran-check-all-num-for-matching-do`
: Fortran 77 では、番号つきの ‘`do`’命令は、それにマッチする行番号をもつ任意の命令で終了します。この目的のためには ‘`continue`’命令を使うのが一般的です (が、強制ではありません)。この変数が非`nil`値の場合、番号が付与された命令をインデントするとき、そこで終了する ‘`do`’をチェックしなければなりません。‘`do`’命令を常に ‘`continue`’(またはよりモダンな ‘`enddo`’) で終了する場合は、この変数を `nil`(デフォルト) にセットすることにより、インデントの速度を上げることができます。

`fortran-blink-matching-if`
: この変数が `t`の場合、‘`endif`’(または ‘`enddo`’) 命令のインデントにより、マッチする ‘`if`’(または ‘`do`’) 命令にカーソルが数瞬移動します。デフォルトは `nil`です。

`fortran-minimum-statement-indent-fixed`
: 固定形式の継続行スタイルを使用する場合の、Fortran 命令にたいする最小のインデントです。命令本体はこれより小さい値でインデントされることはありません。デフォルトは 6 です。

`fortran-minimum-statement-indent-tab`
: タブ形式の継続行スタイルを使用する場合の、Fortran 命令にたいする最小のインデントです。命令本体はこれより小さい値でインデントされることはありません。デフォルトは 8 です。

以下のセクションでは、コメントのインデントを制御する変数を説明します。

Fortran のコメント

通常の Emacs のコメントコマンドは、コード行の後にコメントを記述できると仮定します。Fortran 77 では、標準のコメント構文はコメント行に行全体を要求します。したがって Fortran モードは、標準の Emacs コメントコマンドを置き換え、新しい変数も定義します。

Fortran モードは、‘`!`’で始まり、他のテキストの後に記述することができる、Fortran 90 のコメント構文も処理できます。この構文を許す Fortran 77 コンパイラーは限られているので、Fortran モードは、あらかじめそれを行うように指示しない限り、そのようなコメントを挿入しません。これを行うには、変数 `fortran-comment-line-start`に ‘`"!"`’をセットします。通常とは異なる値を使う場合、`fortran-comment-line-start-skip`も変更する必要があるでしょう。

M-;
: コメントの位置揃え、または新しいコメントを挿入します (`comment-dwim`)。

C-x ;
: 非標準の ‘`!`’だけを適用します (`comment-set-column`)。

C-c ;
: リージョンのすべての行をコメントにします。または (引数を指定した場合は) コメントを実際のコードに戻します (`fortran-comment-region`)。

Fortran モードで実行すると、これは標準の comment-dwimを実行します。これは任意の種類の既存のコメントを認識して、それらのテキストの位置揃えをします。既存のコメントがない場合は、コメントの挿入・位置揃えをします。Fortran モードでのコメントの挿入および位置揃えは、他のモードとは異なります。

新しいコメントが挿入されなければならない場合、カレント行が空のときは、行全体をコメントとして挿入します。その行が空でない場合、もしそれを使うことを指示していれば、非標準の '!'コメントが挿入されます。そうでない場合はカレント行の前に新しい行を挿入して、その行全体をコメントにします。

非標準の '!'コメントは、他の言語のコメントと同じように位置揃えされますが、行全体のコメントは異なります。標準の行全体のコメントは、コメント区切り自体は常に列 0 に出現しなければなりません。位置揃えできるのは、コメントの中のテキストです。変数 fortran-comment-indent-style に、以下の 3 つの値のうち 1 つをセットすることにより、3 つのスタイルの位置揃えを選択できます。

`fixed` テキストを固定列に位置揃えします。これは fortran-comment-line-extra-indent と命令文の最小のインデントとの和です。これがデフォルトです。

最小のインデントは、タブ形式の継続行スタイルの場合は fortran-minimum-statement-indent-tabで、固定形式スタイルの場合は fortran-minimum-statement-indent-fixedです。

`relative` そのテキストがコード行であるかのように位置揃えしますが、fortran-comment-line-extra-indentに指定した列のインデントが追加されます。

`nil` 行全体のコメントを自動的に移動しません。

これらに加えて、変数 fortran-comment-indent-charに、使用したい 1 文字をセットすることにより、行全体のコメントのインデントに使用する文字を指定することができます

コンパイラーにたいする命令行や、プリプロセッサー行は、コメント行と同じ外観をもっています。しかし、fortran-comment-indent-styleの値に関わらず、そのような行が決してインデントされないことが重要です。変数 fortran-directive-reは、どのような行がそのような命令なのかを指定する正規表現です。これにマッチする行はインデントされず、特別な外観のフォントが適用されます。

Emacs の通常のコメントコマンド *C-x ;* (comment-set-column) は再定義されません。'!'コメントを使用している場合、このコマンドをそれらに使用できます。そうでない場合、これは Fortran モードでは役に立ちません。

コマンド *C-c ;* (fortran-comment-region) は、リージョンのすべての行の行頭に文字列 'c$$$' を挿入することにより、これらをコメントにします。数引数を指定した場合、各行の行頭から 'c$$$' を削除することにより、リージョンをコードに戻します。これらのコメントに使用する文字列は、変数 fortran-comment-regionをセットすることにより制御できます。これはコマンドと変数が同じ名前をもつ例であることに注意してください。同じ名前を 2 つの用途で使用することによる衝突はありません。なぜなら Lisp および Emacs ではそれが意味するものは、コンテキストにより明らかだからです。

Fortran モードでの Auto Fill

Fortran モードは、Auto Fill モードにたいする特別なサポートをもっています。これは命令文を挿入するとき、それが長くなりすぎた場合は自動的に分割するマイナーモードです。命令文の分割は、fortran-continuation-stringを使用した継続行により行われます ([ForIndent Cont], page 31 を参照してください)。この分割は SPC、RET、TAB、および Fortran のインデントコマンドにより発

生します。Fortran モードでの Auto Fill の有効化は、通常の方法で行うことができます。Section “Auto Fill” in *the Emacs Manual* を参照してください。

Auto Fill は、その行が望ましい幅 (`fill-column`の値) より長くなった場合は、スペースおよび区切り文字で行をブレークします。Auto Fill が行をブレークする (空白文字以外の) 区切り文字は ‘`+`’、‘`-`’、‘`/`’、‘`*`’、‘`=`’、‘`<`’、‘`>`’、‘`,`’です。`fortran-break-before-delimiters`が`nil`の場合、区切り文字の後ろで行ブレークします。そうでない場合 (デフォルト)、区切り文字の前で行ブレークします。

すべての Fortran バッファーで Auto Fill を有効にするには、`fortran-mode-hook`に`auto-fill-mode`を追加します。Section “Hooks” in *the Emacs Manual* を参照してください。

Fortran での列のチェック

標準の Fortran 77 では、72 列目以降は無視されます。ほとんどのコンパイラーはこれを変更するオプションを提供します (たとえば gfortran の ‘`-ffixed-line-length-N`’)。変数 `fortran-line-length`をカスタマイズすることにより、Fortran モードでの行の長さを変更できます。このポイント以降はコメントに font-lock されます (ただし文字列内の場合は除きます。`fortran-line-length`を超える文字列は、font-lock を混乱させるでしょう)。

C-c C-r　カレント行の上に、“列目盛 (column ruler)” を一時的に表示します。

C-c C-w　`fortran-line-length`列の幅になるように、カレントウィンドウを水平に分割します (`fortran-window-create-momentarily`)。これは、Fortran コンパイラーにより課せられた制限を超えないようにする助けになるでしょう。

C-u C-c C-w
(`fortran-window-create`) 列の幅になるように、カレントウィンドウを水平に分割します。その後は編集を続行できます。

M-x fortran-strip-sequence-nos
列 `fortran-line-length`以上のすべてのテキストを削除します。

コマンド *C-c C-r* (`fortran-column-ruler`) は、カレント行の上に列目盛を一時的に表示します。列目盛は 2 行のテキストで、Fortran プログラムにおいて特別な意味をもつ列の位置を表示します。角カッコ (square brackets) は行番号の範囲を示し、中カッコ (curly brackets) は命令文本体の範囲を示します。列番号がその上に表示されます。

GNU Emacs では、列番号は常に 0 から数えることに注意してください。結果として、この番号はあなたが親しんでいる番号より 1 小さくなるかもしれません。しかしこの行で示される位置は、Fortran の標準です。

列目盛を表示するのに使用されるテキストは、変数 `indent-tabs-mode`の値に依存します。`indent-tabs-mode`が `nil`の場合、変数 `fortran-column-ruler-fixed`の値が列目盛として使用されます。それ以外は、変数 `fortran-column-ruler-tab`の値が表示されます。これらの値を変更することにより、表示される列目盛を変更できます。

C-c C-w (`fortran-window-create-momentarily`) で、一時的にカレントウィンドウを水平方向に分割して、ウィンドウの幅を `fortran-line-length`列にすることにより、長くなりすぎた行を見つけることができます。スペースをタイプすると元の幅に戻ります。

適切な位置でウィンドウを水平方向に分割して、編集を継続することもできます。これを行うには、*C-u C-c C-w* (`M-x fortran-window-create`) を使用します。このウィンドウで編集することより、Fortran での正しい長さを超える行をすぐに見つけることができます。

コマンド *M-x fortran-strip-sequence-nos*は、カレントバッファーのすべての行にたいして、列 `fortran-line-length`以上のテキストすべてを削除します。これは古いシーケンス番号を削除する一番簡単な方法です。

Fortran キーワードの Abbrev

Fortran モードは、一般的なキーワードや定義にたいする abbrev(abbreviation: 省略形) を提供します。あなたが定義できる abbrev と同様なものがあります。これらを使用するには、Abbrev モードをオンに切り替えなければなりません。Section “Abbrevs” in *the Emacs Manual* を参照してください。

ビルトインの abbrev は、1 つの点で特異です。これらはすべてセミコロンから始まります。たとえば Fortran のビルトインの abbrev である ‘`;c`’は、‘`continue`’にたいする省略形です。‘`;c`’を挿入してから、スペースや改行のような区切りとなる文字を挿入すると、Abbrev モードが有効な場合、‘`;c`’は自動的に ‘`continue`’に展開されます。

‘`;?`’または ‘`;C-h`’とタイプすると、すべてのビルトインの Fortran の abbrev のリストと、それが何を意味するかが表示されます。

Emacs と MS-DOS

このセクションでは、Emacs を MS-DOS“オペレーティングシステム”(“MS-DOG” としても知られています) で使用する際の特質を、簡単に説明します。Emacs と Microsoft の現在のオペレーティングシステムの Windows(“Losedows” としても知られています) についての情報は、Emacs のメインマニュアル (Section “Microsoft Windows” in *the Emacs Manual* を参照してください) の中にあります。

Emacs を MS-DOS にたいしてビルドした場合、そのバイナリーは Windows 3.X、Windows NT、Windows 9X/ME、Windows 2000/XP、または OS/2 でも、DOS アプリケーションとして実行されます。MS-DOS にたいしてビルドされた Emacs を使用する場合、このチャプターの内容は、それらすべてのシステムに適用されます。

MS-DOS(および Windows) でのテキストファイルにたいする Emacs の特別な処理については、Section “Text and Binary” in *the Emacs Manual* を参照してください。。

MS-DOS でのキーボードの使用方法

Emacs で `DEL`と呼ばれるキー (ほとんどのワークステーションでそれが指定されているのが由来です) は、PC では `BS`(バックスペース) として知られています。PC 固有の端末の初期化で、`BS`が `DEL`として動作するよう再マップされるのは、これが理由です。同じ理由により、`Delete`キーは *`C-d`*として動作するように、再マップされます。

MS-DOS にたいしてビルドされた Emacs は、*`C-Break`*を *`C-g`*のような、中止 (quit) 文字として認識します。新たな入力にたいして準備ができるまで、*`C-g`*をタイプしても Emacs が検知できないのは、これが理由です。そのため、実行中のコマンドを停止させるために、*`C-g`*を使用することはできません (Section “Quitting” in *the Emacs Manual* を参照してください)。対照的に *`C-Break`*は、(他のシステムでの *`C-g`*のように)、タイプされるとすぐに検知されるので、実行中のコマンドを停止したり、緊急エスケープのために使用されます (Section “Emergency Escape” in *the Emacs Manual* を参照してください)。

PC のキーボードマップは、左 `Alt`キーを `META`キーとして使用します。`SUPER`キーと `HYPER`キーをエミュレートするために、2 つの選択肢があります。変数 `dos-hyper-key`および `dos-super-key`に、1 または 2 をセットすることにより、右 `Ctrl`キーと右 `Alt`キーのどちらかを選択します。`dos-super-key`と `dos-hyper-key`がどちらも 1 以外の場合、デフォルトにより右 `Alt`キーも `META`キーにマップされます。しかし MS^^ef^^bd^^b0DOS の国際化キーボードサポートプログラム `KEYB.COM`がインストールされている場合、非 US 配列のキーボードでは右 `Alt`は`~`や`@`のような文字を入力するために使用されるので、Emacs は右 `Alt`を `META`にマップしません。この場合、左 `Alt`キーだけを `META`キーとして使用することになるでしょう。

変数 `dos-keypad-mode`は、テンキーにより返されるキーコードを制御するフラグ変数です。以下の行を`_emacs`ファイルに記述して、テンキーの `ENTER`キーを、*`C-j`*のように定義することもできます:

```
;; Make the ENTER key from the numeric keypad act as C-j.
(define-key function-key-map [kp-enter] [?\C-j])
```

MS-DOS でびマウスの使用方法

MS-DOS の Emacs はマウスをサポートします (デフォルト端末のみ)。メニューやメニューバーの使用を含めて、マウスコマンドはドキュメントされているように機能します (Section “Menu Bar” in *the Emacs Manual* を参照してください)。MS-DOS の Emacs ではスクロールバーは機能しません。PC マウスには通常 2 つしかボタンがありません。これらは *`Mouse-1`*、*`Mouse-2`*として機能し

ますが、これらのボタンを一緒に押すと、*Mouse-3*の効果をもちます。マウスにボタンが 3 つある場合、Emacs は開始時にそれを検知し、X のようにすべての 3 ボタンは通常のように機能します。

メニューアイテムの上にマウスポインターが移動すると、メニューバーとポップアップメニューにたいするヘルプ文字列が、エコーエリアに表示されます。マウスに反応するテキストはハイライト (Section “Mouse References” in *the Emacs Manual* を参照してください) もサポートされます。

マウスドライバーのいくつかのバージョンは、マウスのボタン数を正しく報告しません。たとえばホイールつきのマウスは 3 つボタンがあると報告されますが、Emacs に渡されるのはそのうち 2 つだけです。真ん中のボタンとして使用されるホイールのクリックも渡されません。このような場合、マウスボタンがいくつあるか Emacs に指示するために、*M-x msdos-set-mouse-buttons*コマンドを使用できます。init ファイル`_emacs`に以下の行を追加することにより、そのようなセッティングを永続化できます:

```
;; Treat the mouse like a 2-button mouse.
(msdos-set-mouse-buttons 2)
```

MS-DOS にたいしてビルドされた Emacs は、Windows 上で実行されているときは、クリップボード操作をサポートします。kill リングにテキストを置くコマンド、または kill リングからテキストを yank するコマンドは、Emacs が X ウィンドウシステムで行なうように、最初に Windows のクリップボードをチェックします (Section “Mouse Commands” in *the Emacs Manual* を参照してください)。Windows 上での MS-DOS 版の Emacs は、プライマリー選択とカットバッファーだけをサポートします。セカンダリー選択は常に空になります。

クリップボードに対するアクセス方法は Windows により実装されているため、クリップボードに置くことができるテキストの長さは、Emacs が利用可能な DOS メモリー量により制限されます。通常はクリップボードに最大 620KB のテキストを置くことができますが、この制限はシステム設定に依存し、Emacs を他のプログラムのサブプロセスとして実行している場合は、もっと少なくなります。kill したテキストが一致しない場合、Emacs はその旨を告げるメッセージを出力して、クリップボードにテキストを置きません。

ヌル文字を Windows クリップボードに置くこともできません。kill されたテキストにヌル文字が含まれる場合、Emacs はそのようなテキストをクリップボードに置かず、その結果にたいするメッセージをエコーエリアに表示します。

変数`dos-display-scancodes`が非`nil`の場合、Emacs は各キーストロークの ASCII値とキーボードのスキャンコードを表示します。この機能は、デバッグのための`view-lossage`コマンドを補足するためのものです。

MS-DOS での表示

MS-DOS のディスプレーでは、bold や italic のようなフォントの変種が使用できませんが、複数のフェイスをサポートしており、それぞれのフェイスでフォアグラウンドとバックグラウンドのカラーを指定できます。したがって m 異なるカラーを使用するために関連するフェイスを定義することにより、フォントを使用する Emacs パッケージ (`font-lock`や Enriched Text モードなそ) の完全な機能を使用することができます。利用できるカラーとフェイスと、それらの外観を確認するには、`list-colors-display`コマンド (Section “Colors” in *the Emacs Manual* を参照してください)、および`list-faces-display` (Section “Faces” in *the Emacs Manual* を参照してください) を使用してください。

DOS ディスプレーでネイティブにサポートされていないグリフと文字を Emacs が表示する方法については、このチャプターの後のほうの、[MS-DOS and MULE], page 41 を参照してください。

Emacs を開始したとき、Emacs はカーソルの形状を塗りつぶしたボックスに変更します。他のシステムではボックスカーソルが Emacs のデフォルトなので、これは互換性のためです。デフォルト

のカーソル形状は、変数 `default-frame-alist`の中の `cursor-type`パラメーターで bar を指定することにより変更できます (Section “Creating Frames” in *the Emacs Manual* を参照してください)。MS-DOS 端末は垂直バーのカーソルをサポートしないので、カーソルは水平バーになり、フレームのパラメターで *`width`*パラメーターが指定された場合、それは実際には水平バーの高さになります。この理由により、MS-DOS ではカーソルタイプ `bar`と `hbar`は同じ効果を生みます。拡張として、以下のようにして、バーカーソル指定には width と同様に、カーソルが行を読み取る開始位置を含めることができます:

```
'(cursor-type bar width . start)
```

これに加えて、*width*パラメーターが負の場合、カーソルバーはその文字セルの最上部から開始されます。

MS-DOS 端末は 1 度に 1 つのフレームだけを表示できます。MS-DOS で動作する Emacs のフレーム機能は、Emacs がテキスト端末で動作する場合と同じように機能します (Section “Frames” in *the Emacs Manual* を参照してください)。MS-Windows で DOS 窓から Emacs を実行した場合、フルスクリーンより小さい可視フレームを作成できますが、それでも Emacs は 1 度に 1 つのフレームしか表示できません。

`dos-mode4350`コマンドはディスプレーを 43 行または 50 行に切り替え、それはハードウェアーに依存します。`dos-mode25`コマンドはスクリーンサイズをデフォルトの 80x25 に切り替えます。

デフォルトでは Emacs が理解するするスクリーンサイズは、列を 80 列、行を 25、28、35、40、43、50 行にセットする方法だけです。しかしビデオアダプターが、ディスプレーを他のサイズに切り替える特別なビデオモードをもつ場合、Emacs もそれをサポートするようにできます。Emacs にフレームを *n*行 *m*列のサイズに切り替えるように指示した場合、Emacs は `screen-dimensions-`*`nxm`* という名前があるかチェックして、もしあれば切り替えるビデオモードの値 (整数でなければなりません) としてそれを使用します (Emacs は `AL`レジスターに `screen-dimensions-`*`nxm`*の値をセットして、BIOS 関数の `Set Video Mode`を呼び出すことによりそのビデオモードに切り替えます)。たとえばビデオアダプターがビデオモードを 85 にしたとき、サイズ 66x80 に切り替わるとしましょう。その場合、以下を`_emacs`ファイルに記述して、Emacs にそれをサポートさせることができます:

```
(setq screen-dimensions-66x80 85)
```

MS-DOS の Emacs は特定のサポートされたフレームサイズだけしかセットできないので、可能性のあるすべてのフレームのサイズ変更要求に従うことはできません。サポートされていないサイズが要求された場合、Emacs は指定されたサイズを越える、次に大きなサポートされたサイズを選択します。この場合、たとえば 36x80 フレームを要求して、かわりに 40x80 を得ることになります。

変数 `screen-dimensions-`*`nxm`*は、指定されたサイズに正確にマッチするときだけ使用され、サポートされた次に大きなサイズを検索するときには、無視されます。上記の例では VGA は 38x80 のサイズをサポートし、`screen-dimensions-38x80`を適切な値で定義していても、36x80 フレームを要求すると 40x80 のスクリーンになります。この場合サイズを 38x80 にするには、`screen-dimensions-36x80`という名前の変数に、`screen-dimensions-38x80`と同じビデオもーどの値をセットして、これを行なうことができます。

MS-DOS でフレームサイズを変更すると、他のすべてのフレームも新しいサイズに変更されます。

MS-DOS でのファイル名

MS-DOS では、ファイル名は大文字小文字を区別せず 8 文字に制限され、それに加えてオプションでピリオドと追加の 3 文字を使用できます。Emacs は他のオペレーティングシステムで、ファイル名を処理するためのこれらの制限を充分認識しています。たとえばファイル名の前のドット ‘.’は MS-DOS では無効なので、Emacs はそれらを透過的にアンダースコアー ‘_’に変換します。したがって、MS-DOS

ではデフォルトの init ファイル (Section “Init File” in *the Emacs Manual* を参照してください) は`_emacs`と呼ばれます。ピリオドの前後の余分な文字は、一般的に MS-DOS 自身により無視されます。したがってファイル`LongFileName.EvenLongerExtension`を visit した場合、それは暗黙に`longfile.eve`となりますが、それでも Emacs はモードラインに長いファイル名を表示し続けます。それ以外では MS-DOS で有効なファイル名を指定するのはユーザーの責任です。上記の透過的な変換は、Emacs に組み込まれたファイル名だけにたいして機能します。

MS-DOS でのファイル名にたいする上記の制限は、オリジナルのファイル名の文字を失うことなしにバックアップファイルの名前を構築するのを、ほとんど不可能にします (Section “Backup Names” in *the Emacs Manual* を参照してください)。たとえば`docs.txt`というファイルにたいするバックアップファイルの名前は、単一のバックアップを使用しているときでさえ`docs.tx~`になります。

Windows 9X、Windows ME、Windows 2000/XP で Emacs を DOS アプリケーションとして実行する場合、長いファイル名のサポートをオンに切り替えることができます。これを行なうと、Emacs はファイル名を切り詰めたり、ファイル名を小文字に変換するかわりに、指定された文字通りのファイル名を使用します。長いファイル名のサポートを有効にするには、Emacs を開始する前に、環境変数`LFN`を‘`y`’にセットします。残念なことに Windows NT は DOS プログラムが長いファイル名にアクセスすることを許さないので、MS-DOS にたいしてビルドされた Emacs は、短い 8+3 のエイリアスだけを見ることになります。

MS-DOS にはホームディレクトリーという概念がないので、MS-DOS 上の Emacs は Emacs がインストールされた場所が、環境変数`HOME`の値であるかのように振る舞います。つまり Emacs のバイナリー`emacs.exe`がディレクトリー`c:/utils/emacs/bin`にある場合、Emacs は`HOME`が‘`c:/utils/emacs`’にセットされているかのように動作します。この場所は特に、Emacs が init ファイル`_emacs`を探す場所でもあります。これを念頭におけば、GNU や Unix のように、ファイル名の中で‘`~`’をホームディレクトリーのエイリアスとして使用できます。Emacs を開始する前に、その環境で`HOME`変数をセットすることもできます。この変数の値は、上記のデフォルトの振る舞いをオーバーライドします。

MS-DOS の Emacs は、`/dev`というディレクトリー名を特別に使います。なぜなら GJGPP のエミュレーターライブラリーの機能は、I/O デバイスの名前がそのディレクトリーにあるかのように振る舞うからです。わたしたちは任意のディスクにたいして、`/dev`という名前のディレクトリーの使用を避けることを推奨します。

印刷と MS-DOS

`lpr-buffer` (Section “Printing” in *the Emacs Manual* を参照してください) や、`ps-print-buffer` (Section “PostScript” in *the Emacs Manual* を参照してください) のようなコマンドは、Posix スタイルの`lpr`プログラムが利用できない場合、出力を 1 つのプリンターポートに送ることにより、MS-DOS で機能します。同じ Emacs 変数がすべてのシステムでの印刷を制御しますが、MS-DOS では異なるデフォルト値をもつ場合もあります。

ネットワークプリンターでの印刷のセットアップに関する詳細は、Section “Windows Printing” in *the Emacs Manual* を参照してください。

プリンターが同じ locale にたいして異なるエンコーディングを使用する Windows 機に接続されている場合にも、非 ASCII テキストの DOS コードページによるエンコーディングを期待するプリンターがいくつかあります。たとえば locale が Latin-1 のとき、Windows はコードページ 1252 を使用しますが、DOS はコードページ 850 を使用します。[MS-DOS and MULE], page 41 を参照してください。Windows からそのようなプリンターで印刷する場合、*`M-x lpr-buffer`*の前に、*`C-x RET c`* (`universal-coding-system-argument`) を使用することができます。その場合、Emacs

は指定した DOS コードページにテキストを変換します。たとえば *C-x RET c cp850-dos RET M-x lpr-region RET*は、リージョンをコードページ 850 のエンコーディングに変換して印刷します。

MS-DOS では後方互換のため、`dos-printer` (`dos-ps-printer`) に値がセットされている場合、`printer-name` (`ps-printer-name`) の値をオーバーライドします。

MS-DOS での国際化サポート

MS-DOS の Emacs は、異なる文字セット同士を変換するためのコーディングシステムを含む、GNU、Unix、その他のプラットフォームでサポートされているのと同じ国際化文字セットをサポートします (Section “International” in *the Emacs Manual* を参照してください)。しかし MS-DOS と、MS-Windows や他のシステムとの間の非互換により、このサポートには知っておくべきいくつかの DOS 特有の状況があります。このセクションではこれらの状況について説明します。

以下の説明では、主に Emacs の MS-DOS ポートについて、特に経験豊富な Emacs ユーザーにとって密接に関係する部分について説明します。

M-x dos-codepage-setup
: カレント DOS コードページにたいして、適切な Emacs ディスプレーとコーディングシステムをセットアップします。

MS-DOS は常に 256 文字の文字セットをサポートするようにデザインされていますが、それからさまざまな文字セットを選択できます。選択できる文字セットは *DOS* コードページとして知られます。各コードページはすべて 128 文字の ASCII文字を含みますが、それ以外の 128 文字 (コード 128 から 255) は、コードページごとに異なります。各コードページは 850、862 のように 3 桁の数字で識別されます。

同時に複数のフォントを使用できる X とは対照的に、通常 MS-DOS は 1 つのセッションで複数のコードページを使用できません。MS-DOS はシステムの開始時に 1 つのコードページをロードするようにデザインされており、それを変更するには再起動が必要です [1]。MS-Windows のような他のシステムで DOS の実行可能ファイルを実行するときも、ほぼ同じ制限が適用されます。

MS-DOS でのマルチバイト処理にたいして、Emacs は選択された DOS コードページで表示できる文字を知る必要があります。そのため起動後に、選択されたコードページ番号を得るためにシステムに問い合わせを行い、その番号を変数 `dos-codepage`に格納します。実際のコードページは異なっていても、カレントコードページにたいしてデフォルト値 437 を返すシステムがいくつかあります (通常これはディスプレーハードウェアーに組み込まれているコードページを使用しているとき発生します)。init ファイルで変数 `dos-codepage`をセットすることにより、Emacs に別のコードページを指定できます。

マルチバイトの Emacs は特定の DOS コードページ — 日本語コードページ 932 のような極東アジアのスクリプトを表示できるものや、1 つの ISO 8859 文字セットをエンコードするものがあります。

極東アジアのコードページは、それらの国々にたいする MULE 文字セットの 1 つを直接表示できるので、Emacs はそのコードページでサポートされる適切な端末コーディングシステムを使用するためにセットアップを行なうだけです。このセクションの残りの部分で説明する特別な機能は、主に ISO 8859 文字セットをエンコードするコードページに関するものです。

[1] 通常 1 つの特定のコードページがディスプレーメモリーに組み込まれていて、`CONFIG.SYS`のようなシステム設定ファイルを変更して再起動することにより他のコードページをインストールできます。再起動なしでコードページを変更できるサードパーティーのソフトウェアーもありますが、ここでは普通の MS-DOS システムが振る舞う方法を説明します。

ISO 文字セットの 1 つに対応するコードページにたいして、Emacs はそのコードページ番号にもとづいた文字セットを認識します。Emacs は、カレントコードページを使用したファイルの読み書きをサポートするためのコーディングシステムを自動的に作成して、そのコーディングシステムをデフォルトとして使用します。このコーディングシステムの名前は `cp`*nnn*で、*nnn*はコードページ番号です。[2]

`cp`*nnn*というコーディングシステムはすべて、モードラインのニーモニックに文字 '`D`' ("DOS") を使用します。端末のコーディングシステムと、ファイル I/O にたいするデフォルトのコーディングシステムは、開始時に適切な `cp`*nnn*コーディングシステムにセットされているので、普通は MS-DOS のモードラインは '`-DD\-`'で始まります。Section "Mode Line" in *the Emacs Manual* を参照してください。極東アジアの DOS 端末は `cp`*nnn*コーディングシステムを使用しないので、Emacs デフォルトのモードラインが初期表示されます。

コードページ番号は使用しているスクリプトも示すので、Emacs はそのスクリプトにたいする言語環境を選択するために、自動的に `set-language-environment`を実行します (Section "Language Environments" in *the Emacs Manual* を参照してください)。

バッファーに ISO 8859 文字セット以外の文字が含まれていて、それが選択された DOS コードページでサポートされていない場合、Emacs は ASCII文字のシーケンスを使用して、それを表示します。たとえばカレントコードページが文字 '`à`'(grave accent つきの小文字の '`o`') にたいするグリフをもたない場合、その文字は '`{`o`}`'と表示されます。ここで中カッコ (braces) はそれが 1 つの文字であることを示す指標です (これはギリシャ文字やヘブライのアルファベットのような非ラテン文字にたいして不格好に見えるかもしれませんが、その言語を知る人はこれを読むことができます)。その文字がスクリーンの複数列を占めていても、それは単なる 1 つの文字であり、Emacs コマンドは、それを 1 文字として扱います。

MS-Windows は独自のコードページを提供し、同じロケールにたいする DOS コードページとは異なります。たとえば DOS コードページと同じ文字をサポートする Windows コードページは 1252 で、DOS コードページ 855 と同じ文字をサポートする Windows コードページは 1252、などです。Emacs の MS-Windows バージョンを '`-nw`'オプションで呼び出したとき、Emacs はカレントコードページを使用して表示を行ないます。

MS-DOS でのサブプロセス

MS-DOS は単一プロセスの "オペレーティングシステム" なので、非同期サブプロセスは利用できません。特に Shell モードと、その変種は機能しません。非同期サブプロセスを使用する Emacs 機能のほとんどは、Shell モードや GUD を含めて、MS-DOS では動作しません。疑わしいときは、コマンドを実行してみれば、機能しない場合は非同期プロセスがサポートされない旨を告げるメッセージが出力されます。

*`M-x compile`*による Emacs でのコンパイル、*`M-x grep`*によるファイル検索、*`M-x diff`*によるファイル間の相違の表示は、同期的に内部プロセスを実行することにより機能します。これはその内部プロセスが終了するまで、編集を行なうことができないことを意味します。

`ispell`プログラムの同期呼び出しにたいする特別なサポートにより、スペルチェックも機能します。これは他のプラットフォームでの非同期呼び出しより遅くなります。

MS-DOS では、機能しない Shell モードのかわりに、*`M-x eshell`*コマンドを使用することができます。これは Posix-like なシェルを、Emacs Lisp で実装した Eshell パッケージを呼び出します。

[2] ISO 8859 にたいす Emacs の標準コーディングシステムは、この目的に完全に沿っているとは言えません。なぜなら DOS コードページは通常、標準 ISO 文字コードにマッチしないからです。たとえば文字 '`ç`'(cedilla つきの '`c`') は標準 Latin-1 文字セットのコード 231 ですが、それに対応する DOS コードページ 850 はこのグリフにコード 135 を使用します。

対照的に、ネイティブな Windows アプリケーションとしてコンパイルされた Emacs は、非同期サブプロセスをサポートします Section “Windows Processes” in *the Emacs Manual* を参照してください。

`lpr-buffer` (Section “Printing” in *the Emacs Manual*) と、`ps-print-buffer` (Section “PostScript” in *the Emacs Manual* を参照してください) は、プリンターポートの 1 つに出力を送ることにより、MS-DOS でも機能します。Section “MS-DOS Printing” in *the Emacs Manual* を参照してください。

MS-DOS でサブプロセスを同期実行する場合は、そのプログラムが終了することと、そのプログラムがキーボード入力の読み取りを試みないことを確認してください。プログラムが自分で終了しない場合、それを終了させることはできません。なぜなら MS-DOS はプロセスを終了させる一般的な方法を提供しないからです。このような場合、*C-c*や *C-Break*を押すことが助けになる場合もあります。

MS-DOS では、他のマシンにあるファイルへのアクセスもサポートされません。何らかのネットワークリダイレクト処理により、MS-DOS にネットワークアクセス機能が組み込まれていない限り、メール送信、ウェブ閲覧、リモートログインなどのようなネットワーク指向のコマンドは機能しません。

MS-DOS の Dired は `ls-lisp`パッケージを使用します (Section “ls in Lisp” in *the Emacs Manual* を参照してください)。したがって MS-DOS の Dired は、変数 `dired-listing-switches` に記述できる、利用可能なオプションは限られます。機能するオプションは ‘`-A`’、‘`-a`’、‘`-c`’、‘`-i`’、‘`-r`’、‘`-S`’、‘`-s`’、‘`-t`’、‘`-u`’です。

Appendix A GNU Free Documentation License

Version 1.3, 3 November 2008

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document *free* in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or non-commercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released

under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

   If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

   If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

   If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

   It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

   You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

   A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any,

be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.

B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.

C. State on the Title page the name of the publisher of the Modified Version, as the publisher.

D. Preserve all the copyright notices of the Document.

E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.

F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.

G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.

H. Include an unaltered copy of this License.

I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.

K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.

L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.

M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.

O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their

titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

   However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

   Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

   Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See `http://www.gnu.org/copyleft/`.

    Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING

    "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

    "CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

    "Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

    An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

    The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts.  A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with. . . Texts." line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

V

W

Y