日英2ヶ国語Emacs音声化システムBEPによるLinuxの音声利用

Development of BEP, Emacs with Japanese-English Text-to-speech Feedback, and Use of Linux by the Japanese Visually Impaired

渡辺 隆行 [1)]　　切明 政憲 [2)]
Takayuki WATANABE　Masanori KIRIAKE

1) 東京女子大学現代文化学部コミュニケーション学科
（〒167-8585 東京都杉並区善福寺 2-6-1 E-mail: nabe@twcu.ac.jp ）

2) ARGV (Accessibility Reseach Group for the Visually-impaired)（ E-mail: seiken@argv.org ）

ABSTRACT. BEP (Bilingual Emacspeak Platform), which adds a speech feedback to GNU Emacs, is a bilingual text-to-speech system for persons with visual disability (mainly for those who are totally blind). BEP automatically switches Japanese text-to-speech engines and English ones depending on the language to be read aloud. In addition to that switching capability, the system can pronounce English with native speakers' pronunciation or with Japanese-English pronunciation, which makes it easier for Japanese users to comprehend English speech output. This paper describes (1) the functions of language identification, (2) the application of proper pronunciation according to the text, and (3) the structure of the bilingual speech server running under Linux.

1 背景

MS-DOS は文字でコマンドを入力し文字で情報が出力される OS であったので視覚障害者にも使いやすい OS であった．1990 年代に入ると初心者にも使いやすい GUI (Graphical User Interface) を持った Windows が登場した．Windows は最新の技術がいち早く導入される反面，ウインドウ・アイコン・メニュー・ポインタなどの GUI 部品による操作を前提にしている場合が多いので Windows 標準の部品を使ったりスクリーンリーダに配慮して作成したりしないと視覚障害者が使えないアプリケーションになってしまう．またバージョンアップの周期が早い巨大な OS に合わせてスクリーンリーダを開発するのは困難な作業である．そのため，Windows はなじみにくいとか使いにくいとか感じる人がいたり，新しいバージョンでは使えない機能が生じたりする．実際，Windows でなくてもできる作業ならば MS-DOS を使うユーザも存在する．

Windows 以外の高機能な OS として，Linux などの UNIX 系 OS が注目を集めている．UNIX はもともとテキストコンソールから操作する OS であり，文字情報だけで OS を管理できる．X Window System などの GUI はテキストの世界の上に別のレイヤとして存在しているだけで，ユーザは好みに応じてテキストツールでも GUI ツールでも UNIX を利用できる．したがって，UNIX も MS-DOS と同じように音声利用できて MS-DOS を利用している視覚障害者が移行しやすい OS であるはずである [1] が，ユーザが少ないこともあって UNIX 用のスクリーンリーダは数少なく，日本語スクリーンリーダで実用化されているものがない．

職場での仕事には正確さと速度を求められるので，仕事で使う音声化システムは情報を効率よく正確に音声化しなければならない．しかし既存の視覚障害者用音声化システムは必ずしもこのような要求を満たしていない．また仕事や勉学の場で出会う情報は日本語と英語が混在している場合が多い．ところが既存の音声化システムは日本語用に設計されているため英語を正しく音声化することが苦手であり，ローマ字を正しく読めるシステムも少ない．したがって，ローマ字や日英 2 ヶ国語が混在した情報を正確に又わかりやすく音声化できるシステムも必要とされている．

2 目的

そこで我々はこのような視覚障害者の要求を満たす音声化システムを開発するために，1999 年に晴眼者と視覚障害者が個人資格で集まったオープンソースのプロジェクトを立ち上げ，日英二ヶ国語の Emacs 音声化システム BEP (Bilingual Emacspeak Platform) の開発に取り組んだ [2].

BEP は Raman 博士が開発した Emacspeak[3] を基にしており，既存の日本語音声化システムと比較して以下の特徴を持つ．

1) GUI に対する AUI(聴覚専用 UI) を持ち，カレンダーのような 2 次元のデータでも聴覚に最適化して情報を提供できる．
2) Emacs が扱う多言語情報を，各言語のテキスト音声合成エンジンで出力できる．
3) 日本人が英語を聞き取りやすいように，ネイティブの英語発音とジャパニーズ英語の発音の 2 種類の発音をコンテキストやユーザの要求に応じて使い分けることができる。
4) バザール型の開発であり，視覚障害者自身が開発の中心にいてユーザが本当に必要とする機能を実装している．また成果をオープンソースとして公開している．
5) Windows でも Linux でも，同じ操作性で Emacs を音声出力で利用できる．

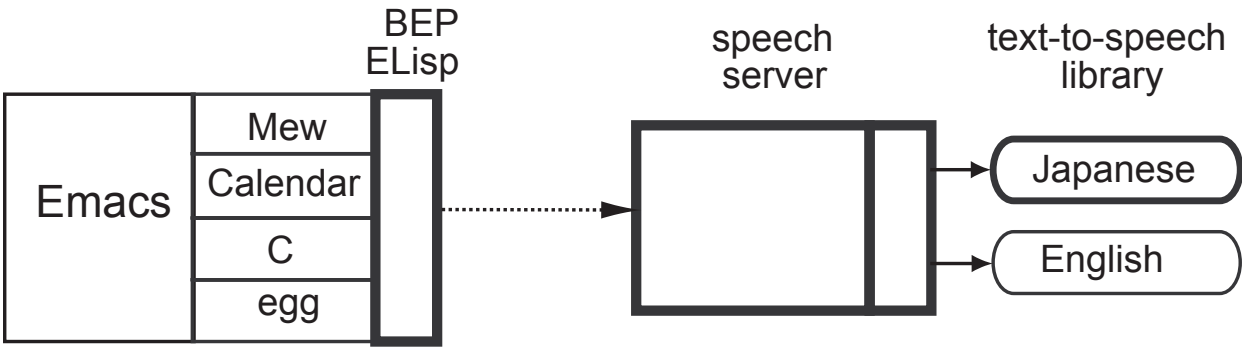

図 1: BEP 構成図

3 BEP の概略

3.1 BEP の構成

図 1 に示すように，BEP は Emacspeak をバイリンガルに拡張した Emacs Lisp (ELisp) 部と，日本語と英語の音声合成エンジンを制御するスピーチサーバ部の 2 つのソフトウェアに分かれる．Emacs Lisp 部は Windows と Linux の両 OS で共通しているが，スピーチサーバは OS ごとに異なる．日本語の音声合成ライブラリは市販の製品を用いている．図中，太線枠部が今回開発した部分．

本論文では，Emacspeak のバイリンガル化と，日英 2 か国語で文字を読み上げることができる Linux 用のスピーチサーバの開発に関して詳しく述べる．

3.2 今回の開発の技術的目標

バイリンガル化した Emacspeak がパソコンを音声利用するユーザに有用で拡張性の高いものになるように，以下の点に留意して開発した．

まず，BEP はバイリンガルシステムである以前に日本語音声化システムとして実用に耐えるものでなければならない．ソースコードフリーで提供されている日本語環境で利用可能な Linux 音声化システムは，2002 年 6 月現在において BEP 以外に存在しない．従って，Linux 上で BEP を使うユーザの多くにとって，BEP がメインの作業環境を音声化する手段となる可能性がある．日本語処理のために十分な機能を提供できなければ，そのような利用に耐えうるものにはならない．このため，BEP では文字の詳細読み，記号類の言語に依存した読み変えなどの機能をサポートした．また，操作のレスポンスを向上するため，音声出力中に後のイベントや操作によって瞬時に先の音声を停止して次のイベントに即した読み上げを開始する即時停止機能にも重点を置いた．

また，上のこととも関係するが，英語の聞き取りが不得意なユーザにも有用なものとすることも重要である．既存の日本のスクリーンリーダーは，英語文字列をカタカナの読みに変換し，日本語音声合成ソフトウェアを用いて発声する．BEP はバイリンガルであるため，英語文字列は英語発音で発声するが，英語が不得意なユーザにとってはカタカナ読みの方が情報を得やすい場面もある．また，日本語文中に現れる英単語などは周囲と同じ日本語音声合成でカタカナ読みするのが望ましい場合が多い．そのため，BEP では英語文字列についてネイティブな読みとカタカナ読みの二つを使い分けることを可能にした．また，日本人の英語聞き取り能力に配慮して，音声出力速度を日英独立に指定できるようにした．

さらに，今回の日英のバイリンガル化の次のステップとして将来的に同じ枠組みで他の言語環境へ拡張可能とすることを考慮し，Emacspeak に英語以外の言語サポートを加えるための拡張，日本語に特化した機能を Lisp ファイルや名前空間のレベルで分離した．

今回の開発成果をより多くの人に利用してもらい，今後も発展していくためには，基盤としている Emacspeak の変更をすぐに取り入れられることが望ましい．そのため，Emacspeak 本体に対する直接の変更を行なわず, コア部分の機能を上位互換で多言語対応のものに再定義する部分と, 多言語対応, 日英バイリンガル特有の機能をモジュールとして別にロードして有効化する形とした．

以下，これらの機能を実現するために Lisp 部及びスピーチサーバ部をどのように構成したかについて述べる．

4 ELisp 部バイリンガル化の機能

4.1 プロパティを利用した言語切り替え

Emacspeak の Lisp 部は大きく二つの部分から構成されている．一つは直接スピーチサーバ (SS) を制御する SS 制御部，もう一つは Emacs の動作 (関数) をトリガとして呼び出され，必要な情報を SS 制御部のために準備するフロントエンド部である．

Emacspeak には情報の種類によって声の種類やパラメータを変えて出力する機能がある．たとえば，プログラムを読み上げさせると，コメントの部分は抑揚のない棒読みで，文字列定数は女性の声でといったように区別される．この機能を Voice-lock と呼んでいる．バイリンガル拡張の実現ではこの Voice-lock にヒントを得ている．

Voice-lock ではフロントエンド部でバッファの内容を解析し，デフォルト以外の声で出力したい部分のテキストに personality 属性を与える．SS 制御部ではこの personality 属性を検知し，スピーチサーバに送るコマンドに声の種類を変更するための in-text コマンドを挿入して声を変化させる．

今回開発したバイリンガル化拡張では，フロントエンド部を拡張し，バッファの内容を見て，文字種や周囲の状況から判断して emacspeak-language 属性をセットする．この属性には値として言語名 (現在は en または ja) がセットされる．次に，SS 制御部ではこの属性の変化するところでスピーチサーバに対して言語変更を示すコマンドを送信することで，その前後で出力すべき言語が異なることをスピーチサーバに伝えることができる．

つまり，一度 emacspeak-language 属性が付与されたテキストは，これまでの音声種別による情報提示と直行する形で言語種別の概念を持つことになる．

4.2 言語属性付与の仕組み

フロントエンド部において常に言語属性を自動的に付与することが困難な場合がある．なぜなら，GNU Emacs が

バッファに保持している各文字の文字セット情報からその部分が属している言語を判定することは自明ではないためである．そのため，BEP の今回の方式では，日本語と英語のみが存在すると仮定した上で言語属性を設定することとした．

言語の判定には主として Emacs の持つ文字カテゴリを用いる．日本語の文字カテゴリ (japanese-jisx0208, katakana-jisx0201 等) に属する文字は日本語としてしか読み上げることができないため，無条件で日本語と判定する．それ以外の文字 (ASCII など) については二つの扱い方が考えられる．一つは英語と判定してオリジナル Emacspeak と同じ処理を行うこと，もう一つは日本語と判定し，カタカナ読み (英単語，ローマ字) や日本語らしい記号読み (半角記号類) で日本語として読み上げることである．

BEP ではバッファの変更，ウインドウのスクロールやリサイズに応じて言語属性を付与する機能が呼び出されるが，言語属性の付与機能を以下の 3 種類の中からユーザの指定に応じて切り替え可能とした．バッファローカル変数とすることで，バッファごと，モードごとに設定することもできる．また，後から追加することも可能である．現在提供している三つの方式を以下に示す．

1) ネイティブ英語モード: ASCII 文字はすべて英語と判定し，英語の音声合成を用いて読み上げる．
2) 混在モード: 一定の長さ (デフォルト 40 文字) 以下の ASCII 文字の連続は日本語と判定し，カタカナ読みで日本語の音声合成に送る．これは，段落全体が ASCII 文字であるような場合には英語で発音してほしいが，日本語文中に現れる短い英文字列は日本語化した語で日本語の一部として読み上げる方が自然と思われる場合が多いためである．このモードをデフォルトとしている．
3) すべてカタカナモード: ASCII 文字列もすべてカタカナ読みとして日本語の音声合成で出力するモードである．プログラムなど特定の環境では日本人のユーザにとってこの方が実用的な場合がある．

4.3 その他 Lisp 部多言語対応について

本プロジェクトにおける Emacspeak に対する拡張は，4 つの部分に分けることができる．それらに関して概略を述べる．

1) Emacspeak 本体の機能拡張: 主にこれまでに述べた言語属性の付与と利用に関する切り口を提供することが目的である．また，言語属性に応じて実際に言語変更をスピーチサーバに通知したり，必要に応じてそれぞれの言語での読み上げ速度を設定する部分は直接スピーチサーバと関わるためにこの部分に含まれる．この部分は Emacspeak の関数を上位互換な関数で再定義することで実現している．
2) 多言語拡張のフレームワーク: 言語に依存しないインターフェースを上記機能拡張部に対して提供する．Emacspeak は機能拡張部を通じてこのレイヤの関数のみを呼び出し，言語に依存した処理への分岐はこのレイヤで行う．抽象化された関数には，主に文字や記号の言語に適した読み方の取得 (例: 「(」を「left paren」と読むか「カッコ」と読むかなど)，カーソル移動時や入力時に適した読み方の取得，言語属性付与のためのラッパーなどが含まれる．
3) 日本語特有の部分: 多言語化フレームワークから呼び出される．ここには日本語として適切な文字の読み方やカーソル移動/日本語変換時のフィードバックに適した読み方の取得，前節で述べた言語属性の付与に関する三つの方式の実装などが含まれる．
4) 日本語環境でよく用いられて Emacspeak がサポートしていない ELisp アプリケーションのサポートファイル群: 日本語変換システムである tamago V4 の音声フィードバックサポート，Mew， emacs-w3m 等のサポートが含まれる．

4.4 ユーザから見える変更

本プロジェクトで開発した Bilingual Emacspeak Platform ではオリジナル Emacspeak の機能を日本語英語の混在環境でもシームレスに利用可能とする方向で拡張を行った．そのためユーザから見える部分での操作的変更は比較的少なくなっている．

以下は多言語拡張に関連した interactive コマンドの一覧である．

emacspeak-m17n-auto-put-language-mode : バッファの変更，ウインドーのスクロール，リサイズなどのイベントに合わせて，表示されている部分に自動的に言語プロパティを付与する機能を on/off する．同名の変数をトグルするとともに，関連するフックの設定と解除を行う．

emacspeak-m17n-ja-change-strategy : 日本語と英語のバイリンガル環境において，利用する put-language-strategy を変更する．詳しくはカスタマイズ変数の項を参照．

emacspeak-m17n-put-language-region : 現在の emacspeak-m17n-put-language-strategy に従って指定した領域に言語プロパティを付与する．

emacspeak-m17n-sync-rate-offset : 言語ごとの発声速度オフセットが指定されている場合に，それをスピーチサーバに反映するコマンドを送信する．

英語が混じった文章を聞き取りやすくするために，BEP はユーザのニーズに応じてバイリンガル情報の発音方法を使い分けることができる．この 3 種（ネイティブ，混在，すべてカタカナ）の発音モードは以下のコマンドで切り替える．

バイリンガルモードの切り替え C-e x m s : 読み上げの方式を変更する．ASCII 文字は総てネイティブ発音，ある一定以上の長さの文字列はネイティブ発音，総てカタカナ英語 (日本語発音の英語) の 3 通りを切り替えることができる．

バイリンガル拡張で導入された，ユーザで設定可能な変数の一覧を示す．

dtk-speaker-process-coding-system : スピーチサーバに送信されるコマンド文字列の coding-system．現在の Windows 及び Linux 用スピーチサーバでは Shift_JIS を用いる．

dtk-default-language : 言語属性が張られていない文字を読むときのデフォルト言語．

emacspeak-m17n-auto-put-language-mode : 自動的に言語属性を付与するかどうかを示す．直接変更せず，emacspeak-m17n-auto-put-language-mode コマンドで変更する．

emacspeak-m17n-put-language-strategy : 言語属性を付与する関数を指定する．指定された関数には，引数として領域の先頭と最後のポイント，変更前の領域の長さが渡される．バッファごとにローカルに変更することができる．直接変更せず，emacspeak-m17n-set-put-language-strategy を用いる．

emacspeak-m17n-put-language-internal-strategy : バッファではなく Emacs が表示するメッセージなど，言語属性が付与されない状態で SS 制御部に直接渡された文字列に適用する属性付与関数を指定する．仮定さ

れる引数は emacspeak-m17n-put-language-strategy と同じ．直接変更せず，emacspeak-m17n-set-put-language-internal-strategy を用いる．

emacspeak-m17n-rate-offset-alist : 言語とそれに対応した速度オフセットの alist． 例: ((ja 0) (en -30))

emacspeak-m17n-ja-strategy-sets : 日本語とのバイリンガル環境で，emacspeak-m17n-ja-change-strategy で選択できる言語属性付与関数のセットとそれに対応する説明文字列，切替えキーを定義する．

emacspeak-m17n-ja-ke-limit : 言語属性付与が混在モードの時，何文字以上の ASCII 文字列を英語として扱うかの指定．デフォルトは 40.

emacspeak-m17n-ja-ke-view : 言語属性付与が混在モードで変更前の長さが ke-limit より短い時，ポイント位置から上下に何行を見て判定するかの指定．デフォルトは 3 で，上下 2 行を見る．

5 Linux スピーチサーバの構成

スピーチサーバは BEP の音声出力を行う独立したプログラムで，Emacs のサブプロセスとして起動される．BEP の SS 制御部からコマンド入力を受け，日英の音声合成 (TTS) ライブラリを制御し，生成された PCM データをサウンドデバイスへ書き込む．また，コマンドに従って出力の即時停止を行う．本節では新たに開発した Linux 用バイリンガルスピーチサーバの構成について解説する．

5.1 制御の仕組み

- Linux 用スピーチサーバは，コマンド処理部，コントローラ部，TTS ラッパー部，TTS ソフトウェア，サウンドデバイス制御部から構成される．
- コマンド処理部は Emacspeak からのコマンドを 1 行ずつ受け取り，コントローラ部のメソッドにマップする役割を持つ．
- コントローラ部はコマンドに応じて，リクエストの作成とキューイング，共通 TTS インターフェースを介した TTS の管理リクエストの送信を行う．
- TTS ラッパー部は TTS ソフトウェアと一対一に対応し，共通 TTS インターフェースを提供する．それぞれがコマンド処理部とは独立したスレッドとして動作する．
- サウンドデバイス制御部はサウンドデバイスへの出力を一元管理し，出力要求の順序を管理する．また，サウンドデバイスドライバの違いを吸収する．

5.2 共通 TTS インターフェース

Linux スピーチサーバで利用する日本語と英語の TTS ソフトウェアは日本語と英語で全く異なる API を備えている．

日本語： クリエートシステム開発株式会社が販売する Linux 用の日本語音声合成ライブラリ dtalker を用いる．日本語文字列を解析して独自の表音文字列に変換する言語処理ライブラリと，表音文字列を入力として PCM データを生成する波形処理部からなる．API は言語処理と波形処理それぞれに対して定義されている．ライブラリ自身は音声をサウンドデバイスへ出力する機能を持たない．音声の種類や速度を変更するには波形処理部のライブラリ関数を呼ぶか，波形処理部に入力する表音文字列に制御文字列を付加する．

英語： IBM の Linux 用音声合成ライブラリである ViaVoice を利用する．文字列の入力，キューイング，波形出力を統合的に制御する Eloquence Command Interface (ECI) と呼ばれる API を持つ．生成した波形をサウンドデバイスに出力する機能を持つ．音声の種類や速度の変更は入力に「‘」で始まるコマンド文字列を挿入することで読み上げ文字列と同期して行うことができる．

このように異なる機能を持つ TTS を制御するため，別に共通の API を定め，それに会わせたラッパークラスを TTS ごとに作成して，上位のレイヤーからは共通 API のみを利用することとした．

共通 TTS インターフェースではそれぞれの TTS が独立したスレッドとして動作する．そのうえで，リクエストのキューイング，音声合成処理，即時停止の機能を提供する．TTS へのリクエストは発声すべき文字列，発声速度，言語，タイプ (文字列として読むか 1 文字として扱うのか)，句読点の読み方に関するパラメータなどを含む．リクエスト中の文字列には，読み上げるべき文字列以外に，TTS ソフトウェアを制御するための in-text コマンドも含まれる．これは DECTalk のサポートするコマンドのサブセットであり，[,] で囲まれる．これらのコマンドはリクエストからそれぞれの TTS ラッパーに渡され，そこで TTS ソフトウェアに適した形式 (別の in-text コマンドまたはライブラリ用 API のコール) に変換する．

共通 TTS インターフェースでは以下の機能を定義する．

- エンジンがサポートする言語の設定と取得
- 発声速度オフセットの指定
- 出力先サウンドデバイスの指定
- TTS スレッドの開始と停止
- 文字列のキューイング
- 発声開始
- 発声停止フラグのセット

TTS のスレッドが開始されるとそれぞれの TTS ソフトウェアを初期化し，内部のキューにリクエストが入力されるのを待つ．上位の制御でリクエストが入力され，発声開始が要求されると，サウンドデバイスの待ち順を予約し，リクエストを順に処理して音声出力を行う．また，発声停止要求があれば，キューイングされたリクエストを破棄してリクエスト待ち状態に移行する．

5.3 コントローラ部と多言語の扱い

コントローラ部は発声リクエストのキュー，使用可能な TTS のリスト，現在の発声パラメータを保持している．それぞれ入力されたコマンドによって設定することができる．パラメータの主なものを以下に示す．

現在の言語 次にコマンドによってキューイングされる文字列を読み上げる言語を指定する．現在は ja, en のみサポート．

発声速度 WPM(Word per Minute) で指定される発声速度．これを元に各 TTS の速度が決定される．

記号類読みモード 記号や句読点の読み上げをどの程度行うか．none, some, all の三つのモードがあり，none では句読点や括弧類などの記号を読み上げず，all ではそれらすべてを正確に発声する．

大文字による複合語分離指定 (split_caps) 英字複合語を大文字のところで区切って解析し読み上げるかどうかの指定．「ssDspDevice」といった記法でも意味が分かるように読み上げることができる．

1 文字読み上げ時の速度倍率 コマンドで 1 文字のみの読み上げ (大半は Emacs へのキー入力操作のエコーバック) が指定されたとき，現在の発声速度にこの倍率をかけた速度にして発声する．通常は 1 より大きい値を指定することでキーエコーバックを高速化する．

コントローラ部は，コマンドとして文字 (列) のキューイングが要求されると，与えられた文字 (列) と上記のパラ

メータをリクエストとしてキューにプッシュする．コマンドとして発声が要求されると，キューから一つずつリクエストをポップして，そこに指定された言語に従ってそれぞれの TTS ラッパーに送る．最後にすべての TTS ラッパーに発声開始を要求することで文字列が順次適切な TTS で読み上げられる．

5.4 Linux スピーチサーバ制御コマンド一覧

Linux スピーチサーバのコマンド処理部が解釈するコマンドの一覧を示す．

q {<string>}： 文字列 string と現在のパラメータをキューに入れる．
d： キューの中身を順に音声化する．
s： 音声化中の音声の即時停止とキューのクリア．
l {<C>}： 1 文字 C を即時発声（大文字の区別，速度加速あり）．
tts_say {<string>}： 文字列 string を即時発声．
t <freq> <msec>： トーンサインをキューに入れる (予約)．
tts_set_speech_rate <NNN>： 音声速度を NNN (WPM) に設定．
tts_set_punctuations <all|some|none>： 記号類の読み上げモードを指定．
tts_set_language <en|ja>： 現在の言語を指定．
tts_set_character_scale <N.N>： 1 コマンドの読み上げ増速倍率を設定．
tts_sync_state (**略**)： このコマンドは <punc-mode> <capitalize> <allcaps-beep> <split-caps> <rate> パラメータを一括設定する．punc-mode は記号類読みモード (all|some|none)，split-caps は大文字による区切り読み指定 (0:off, 1:on)，rate は読み上げ速度 (WPM)．その他は実装していない．
tts_set_speed_offset <lang> <offset>： 言語ごとの速度オフセット値指定．言語 lang(ja|en) の速度オフセットを offset に設定する．−20 と指定すると，その言語による読み上げはその時点での速度値より 20WPM 遅くなる．

6 まとめ

6.1 入手方法及びインストールの手順

当プロジェクトで開発した部分は GNU GPL に従って，BEP の Web サイト（http://www.argv.org/bep/）で公開している．ソースの入手方法及びインストールの詳細は Web を参照されたい．

6.2 動作確認環境

本システムは Windows でも Linux でも使える Emacs の音声化システムであり，今回開発した部分 (Emacs Lisp ライブラリ, 各 OS 用のスピーチサーバ) 以外に, Emacs と日本語と英語の音声合成ライブラリが必要である．Linux 版は以下の環境で動作を確認している．

- CPU: Pentium MMX 266MHz 以上
- Memory: 128MB 以上
- 必要なハードディスク容量: 20MB 程度

以下のソフトが事前にインストールされている必要がある．これらのインストールに約 70MB 程度が必要である．

- Emacs: Emacs 20.6, 20.7 21.1
- 日本語音声合成エンジン: クリエートシステム開発 (株) の Document Talker for Linux
- 英語音声合成エンジン: IBM の Viavoice Outloud

以下のディストリビューションで動作確認を行った．

- RedHat 6.20, 7.2(J)
- Vine Linux 2.1.5

6.3 今後の課題

今回の開発で IPA の申請書に書いた中核部分は完成させることができたが，実際にユーザに使ってもらうためにはまだまだ改良が必要である．バイリンガル化に関してもばまだ拡張が不完全なところがある．本家 Emacspeak へのコードの還元もこれからの課題である．

Emacs は一般ユーザになじみがないアプリケーションであり，Linux 自身も一般には普及していないので，BEP に関心があっても使い始めることができないユーザも存在する．BEP を単なるソフトウェアを超えたひとつのシステムと見た場合，このようなユーザの教育やサポートも今後の大きな課題である．

今回 Linux 版の BEP が動くようになったので，これを組み込み Linux に応用できないかと考えている．つまり，最近は i-mode や PDA などの携帯情報機器の重要性が増しているので，これらの機器を視覚障害者も利用できるようにしておく必要があり，iPAQ や Linux 版 Zaurus などに BEP を組み込んで，視覚障害者も使える携帯情報機器を実現できないか検討している．

6.4 音声合成への要望

本システムを使用するとき，音声合成ライブラリだけがオープンソースなソフトウェアを使用できない．英語の音声合成ライブラリは Windows 用も Linux 用も無料で配布されているものがあるが，日本語に関しては両 OS ともメーカの製品を購入しなければならない．視覚障害者が音声でコンピュータを利用する際，音声の機能や品質は極めて重要な要素であるが，彼らの要求を満たすものはメーカが開発した商品以外にはない．外部から資金でソースのライセンスを購入するなどの方法で，商用品質の日本語音声合成ライブラリをオープンドメインに置けないものかと考えている．

視覚障害者が使用する音声合成は倍速以上の高速で音声化しても聞きやすい声を持ち，しかも応答性がよいなどの機能が必要である．しかし現在の音声合成の研究開発は，「重くてもよいから人間のように自然に発話する」ことに主眼が置かれ，視覚障害者のニーズを満たせていない．

7 参加企業及び機関

本システムの開発は，著者らを含む晴眼者と視覚障害者の有志が個人の資格で取り組んだ．

参考文献

[1] 渡辺隆行, 井上浩一, 坂本貢, 切明政憲, 白藤秀樹, 本多博彦, 西本卓也, 釜江常好: Bilingual Emacspeak と視覚障害者の Linux アクセシビリティ向上, *Linux Conference 2000 Fall*, 京都, pp.246–255 (2000).

[2] Watanabe, T., Inoue, K., Sakamoto, M., Kiriake, M., Honda, H., Nishimoto, T., and Kamae, T.: Bilingual Emacspeak Platform – A universal speech interface with GNU Emacs –, *Proceedings of HCI International 2001* (Stephandis, C. (ed.)), New Orleans, Lawrence Erlbaum Associates, Vol. 3, pp.446–449 (2001).

[3] Raman, T.V: *Auditory user interfaces ー toward the speaking computer ー*, Kluwer Academic Publishers, Boston (1997).