日英2ヶ国語Emacs音声化システム
Bilingual Emacspeak Platform

井上 浩一[A]、切明 政憲[A]、渡辺 隆行[B]

[A]ARGV (Accessibility Research Group for the Visually-Impaired)
[B]東京女子大学現代文化学部コミュニケーション学科
E-mail: [A]{inoue|seiken}@argv.org [B]nabe@twcu.ac.jp

Emacs はエディタであると同時に、Web ブラウザやメーラ、シェルなども含めた統合環境として広く用いられており、Emacs Lisp による拡張性、Linux, Windows を含めた多くのプラットフォーム上で動作するポータビリティを持っている。T. V. Raman はこの Emacs を使って、UNIX を利用する視覚障害者のための統合デスクトップ Emacspeak を開発した。我々はこの Emacspeak に着目し、その機能を拡張して、多言語、特に日英 2 カ国語、そして Linux と Windows に対応した音声化システムとすべく、BEP の開発を進めてきた。本発表ではようやく実用のレベルとして公開可能となった BEP の特徴、機能、構成について述べる。

1 背景

まず本論文の前提となる障害者、特に視覚障害者を取り巻くコンピュータ環境の変遷と現状について概説する。

1.1 支援技術としてのコンピュータ

コンピュータが個人にも入手可能となったのは 1980 年代、パーソナルコンピュータが登場してからのことであるが、当時のパソコンはまだ主に趣味としてのゲームやプログラミングの道具として限られた人に利用されるに留まっていた。しかし、障害者のための支援技術としてコンピュータを用いる試みがこのころから活発に取り組まれはじめていた。視覚障害者に関しては、漢字とかなが混じった文章を自力で書いて印刷する道具として 1983 年に AOK 点字ワープロ[1] が開発され、パソコンの一般への普及を待たずに広まった。また、OS レベルで画面を監視して読み上げるスクリーンリーダ（画面読み上げソフトウェア）[2]が普及すると、上記に加えて健常者との間でのアプリケーションや電子文書の共有、パソコン通信を通じた広範囲なコミュニケーションも行なわれるようになった。

点字出版にも変化が訪れた。点字はそれまで紙に直接記録される文字で、その複製や再利用には大きなコストを必要としたが、パソコン上での編集、点字プリンタでの印刷が一般化し、完全とは言えないものの、電子文書から点字への自動変換も可能となった。点字への翻訳が効率化されることによって、以前よりは省力かつ短期間で点字情報が得られるようになった。

パソコン普及の黎明期、最も実質的に恩恵を受けたのは障害者であったと言っても過言ではないかもしれない。

1.2 GUI の登場と視覚障害者

1990 年代の Windows の登場によって、マウスを用いた、より視覚的で直観的な操作が可能となり、一般の人でもコンピュータを簡単に使えるようになった。文書作成のほとんどはワープロソフトを用いてパソコン上で行なわれるし、インターネットを利用した情報入手や発信は我々の生活にとってなくてはならないものになっている。紙文書中心の時代から電子文書が流通する時代への変化は視覚障害者にとっても喜ばしいことであるが、これにともなって新たな問題も生じてきた。

[1]高知システム開発により PC8801 上で実現されたワープロ。漢字を点字 3 または 4 マスで表す 6 点漢字体系を用いて入力することで、文書を作成し印刷できる。

[2]MS-DOS 用のスクリーンリーダとしては、斎藤正夫氏による VDM100 シリーズ、石川准氏のグラスルーツなどがある。

その一つはユーザインターフェースの変化である。Windows 登場以前に用いられていた MS-DOS は文字ベースのユーザインターフェース（CUI: Character User Interface）を通じて操作していた。CUI では操作に必要な情報は文字であり、この文字情報は 80×25 といったマトリクスの上に提示され、ユーザはキーボードからの文字入力で応答する。こういった文字ベースの入出力は、音声または点字出力で画面を確認し操作する視覚障害者にとって把握しやすく、スクリーンリーダにとっても情報の取得が比較的容易であった。それに対し、Windows などの GUI（Graphical User Interface）では OS によって提示される情報は文字と絵文字、図形の混合したもので、位置情報が重視され、表示された情報は必ずしも音声や点字で表現することに適していない上、ユーザはマウスを用いてしか応答できない場面も多かった。

もう一つは、支援技術開発の遅れである。現在、障害者がコンピュータを利用する場合、障害に応じて利用を支援する補助機器やソフトウェアが必要になり、それらは OS とは独立して開発される。視覚障害者がコンピュータを利用するためには、画面上の情報を音声または点字として出力する手段が必要である。それはスクリーンリーダであったり、特定の用途を対象とする独立したアプリケーションであったりする。これらはその重要性にも関わらず、通常のアプリケーションと同じように特定の基本ソフト（OS）を前提に開発されるのが一般的であり、その性質上、ともすれば他のアプリケーションよりも OS に依存した形で開発せざるを得ない。例えば DOS のスクリーンリーダは画面情報が文字で格納されたメモリ領域に直接アクセスできることを前提として作成される。

これらの事情から、視覚障害者は目前に現れた新たな OS、Windows を実用的に使えるようになるまでに長い期間を要し、世の中を変えたインターネットの普及を始めとする新しい技術が Windows の普及と不可分に進行したため、深刻な状況となっていた [1]。1990 年代後半から日本でもいくつかの Windows 用スクリーンリーダが実用化され[3]、最適とはいえないまでも、Windows を用いてインターネットから情報を入手し、また発信している視覚障害者は多い。しかし、現在でもコンピュータに熟練した視覚障害者の中には、効率がよい、状況が把握しやすいといった理由から、MS-DOS や DOS プロンプトといった CUI を常用する人も少なくない。

1.3 UNIX のアクセシビリティ

UNIX 系 OS の状況はまた異なっている。最近では X Window System などのウインドウシステムが広く利用されているが、OS の持つユーザインターフェースは CUI であるので、文字ベースでほぼすべての操作を行なうことができる。そのため、熟練した視覚障害者の利用に本質的に適していると考えられるが、支援技術の進歩は Windows に比較して進んでいるとはいえない。PC-UNIX が普及して自分のコンピュータにコンソールからログインして利用する形態が一般化したが、視覚障害者の多くは今もスクリーンリーダの動作する DOS マシンから telnet によりログインして利用している。欧米では 1990 年代後半から Linux コンソール用スクリーンリーダが開発されている[4]が日本語では使えない。

また最近では、米国のリハビリテーション法 508 条の改正が追い風となって、Sun Microsystems などの商用 UNIX ベンダーが主導した形で GNOME デスクトップをアクセス可能にするプロジェクトが進行している。これはアクセシビリティの確保に必要な情報の取得や操作を行なう API を定め、それを GNOME を構成するツールキットに実装する形をとっており、その API を利用した GNOME 2.0 用スクリーンリーダ/画面拡大ソフトである Gnopernicus が開発されつつある。

しかしこれらを日本人が使うためにはマルチバイト文字である日本語を扱わねばならず、これらの動きから取り残された状況にある。1999 年にはコンソール用スクリーンリーダ開発が試みられたが[5]、広く公開されないまま開発が終了して現在に至っており、2002 年 8 月現在、BEP 以外には Linux 用日本語音声化システムは存在しない。

[3] システムソリューションセンター栃木の 95Reader、高知システム開発の PC-Talker、欧米のソフトからローカライズされた outSPOKEN、JAWS などがある。

[4] コンソールの文字情報を点字ピンディスプレイに表示する BRLTTY、カーネルパッチの形で画面の読み上げを実現する Speakup などが知られる。

[5] ASUKA（ネットワーク応用通信研究所）が知られている。

2 Bilingual Emacspeak Platform

2.1 BEP の概要

BEP（Bilingual Emacspeak Platform）は後述する Emacspeak をベースにし、日英 2ヶ国語に対応した Emacs 音声化システムである。

GNU Emacs は UNIX を利用する研究者や技術者の間で広く利用され標準となっているエディタである。テキスト編集のみならず、ファイラー、shell、Web ブラウズ、電子メール、ftp、telnet、辞書検索、仮想端末、PIM、カレンダー、日記、プログラム開発（デバッグ）など日常必要とされる多くの作業を Emacs 内で行なうことができる。更に、欧米、アジアなど多くの文字セットを標準でサポートする。

また、UNIX や Windows を始め多くの OS 上で動作する。基本部分以外は Emacs Lisp という言語で書かれ、Lisp プログラムによって柔軟な拡張が可能である上、対応する OS 上ではほとんど同じ Lisp プログラムが動作する。この意味で Emacs は VM（Virtual Machine）であると見ることもできる。

Emacspeak は、自身が全盲の T. V. Raman 博士が開発した Emacs 音声化システムである。Emacs 内部から得られた情報を聴覚に最適化して出力することができ、Linux システムを利用するためのフロントエンドとして利用することを想定して設計されている。

BEP はこれらの点に着目し、視覚障害を持つ開発者を含む Bilingual Emacspeak Project で以下を目的として開発を進めている。

- Linux のための日本語対応フロントエンドの一つを提供する。
- 視覚障害者のための、OS の違いを越えた統合環境を提供する。
- 日本語音声化システムに必要な機能を既存の日本語スクリーンリーダから取り込み、実用的なバイリンガル音声化システムとする。

2.2 Emacspeak

Emacspeak は GNU Emacs を構成する言語である Emacs Lisp で書かれており、Emacs の動作にともなって呼び出される多数の追加機能の集まりである。呼び出された Emacspeak の関数は、Emacs が取り扱う情報を取得して、音声化した時に分かりやすいように情報を加工する。また、インテキストコマンドの形でスピーチサーバへの命令も埋め込む。その後、スピーチサーバ[6]にこれらの文字列を送り出す。この構造によって、スピーチサーバを OS ごとに準備すれば Emacspeak 自体は Emacs と同じポータビリティを持つことになる。Emacs はテキストエディタとして設計されており、Emacs Lisp で情報を自由に加工できる。また高度なカスタマイズも容易である。さらにエディタそのものが音声出力を制御しているため、十分な応答性の確保と、提示情報の詳細なチューニングが可能である。したがって Emacspeak によって、研究・開発・教育等に必要なほとんどの作業が可能となる。なお Emacspeak の詳細な説明は、Emacspeak のユーザマニュアル[7] に書かれている。

Emacspeak は Auditory User Interface （AUI）[2] を使って Emacs を音声化する。音声表示用の AUI は画面表示用の GUI とは独立に情報の論理構造にアクセスして必要な情報を取り出し、聴覚に適した形で音声化する。Emacspeak は buffer の mode に応じた最適な AUI を Emacs Lisp で実装しており、calendar-mode ではカレンダーに適した機能を供給でき、W3-mode では CSS2 （Cascading Style Sheets, level 2）[8] の音声スタイルシートに対応した Web ブラウザとなる。

画面に表示される文字には複数のフォントがあり、さらに太字やイタリックなどの修飾をしたり大きさを変えたりできる。音声においても、ピッチなどの声の個性を変えたり、男女を変えたり、年齢を変えたり、ボリュームを変えたりすることで、テキストの読み上げに対して多様な修飾ができる。Emacspeak には audio formatting や voice-lock という機能があり、プログラムソースを編集するときはコメント、キーワード、関数名、文字列、などの構成要素ごとに異なった声を割り当てたり、大文字の場合は短いビープ音を鳴らして大文字であることを示したり声の調子を変えることで、どこが大文字か耳で聞いただけでわかるようになって

[6] スピーチサーバは、音声合成ハードウェアか、Emacs とは非同期に動作する別プロセスのソフトウェアである。

[7] Texinfo 形式のドキュメントが用意されている。

[8] CSS2 では、画面、音声出力装置、プリンタ、点字出力装置などの異なった出力装置ごとにスタイルシートを指定できる。

いる。

このように Emacspeak は、全盲の大学院生であった Raman 氏が自分自身の博士論文執筆のためにいろいろな人の協力を得て開発したので、本当に必要な機能が実装されている極めて高機能な音声化システムである。しかしながら、Emacs を知らなければ使えない、UNIX 以外の OS を考慮していない、英語以外の言語を全く考慮していないという問題から、日本のユーザが日常生活で利用できるものではなかった。

2.3 BEP の特徴

BEP（Bilingual Emacspeak Project）は 1999 年秋から活動を始め、2001 年 1 月に最初のリリースを公開した。この段階の BEP については Linux Conference 2000 Fall でも発表 [3] したが、このときの BEP は以下の機能を備えていた。

- 日本語を扱える
- 日本語文字（漢字）の説明読みによる識別、日本語入力のサポート
- 応答性に優れ、入力文字に応じてスピーチサーバで音声合成の言語を自動切換えする日英読み分け機能を持つスピーチサーバ（ただし Linux 版は日本語のみ）

  Windows では読み上げる文字列に基づいたバイリンガル音声を実現したが、Linux でのバイリンガルサポート、システムの安定化、本家 Emacspeak 新機能への追従など課題を残していた。その後、平成 13 年度の IPA 未踏ソフトウェアの資金などを得て Lisp 部の内部構造とスピーチサーバの実装を根本的に見直して再開発し、2002 年夏には BEP は上記に加えて以下のような特徴を合わせ持つ音声化システムとして公開できるに至った [4]。

- Emacspeak の持つ機能を継承し、その拡張として動作する。Emacspeak はバッファのモードに即した最適な情報定時、文書やデータ構造に応じて声を変えて表現する voice-lock、文字種（大文字/小文字）識別などの機能を持ち、メーラーや WEB ブラウザなど多くの拡張パッケージに対応している。これらの機能はそのまま BEP で利用できる。また、Emacspeak のプログラム自体には変更を加えず、必要な部分のみ動的に再定義することで本家への追従が容易になった。
- Emacs Lisp 部でコンテキストに基づいて言語を判定し、バイリンガルスピーチサーバを制御する形式の日英 2ヶ国語音声化システムに進化した。また付録 A に示すように、日英 2ヶ国語が混在した情報を読み上げるときには，カタカナ英語とネイティブ英語の 2 種類の英語の発音が必要であることがわかったので、英語はこの 2 つの発音を柔軟に選択可能とした。
- スピーチサーバの改良によって Windows と Linux で同レベルのバイリンガル環境を実現し、スピーチサーバの安定性も高まった。
- 日本でポピュラーな Lisp パッケージ（Mew, Wanderlust, w3m 等）をサポートした。

本システムの設計・開発に当たっては、バイリンガルであること以前に、まず日本語を扱う環境として実用に耐えるものとするように努めた。そのため、初期リリースから実装していた文字の詳細読み、操作による音声即時停止のレスポンスを重視し、改良した。また、記号類の言語に依存した読み変え、カタカナ読みされる文字列内のローマ字を正確に読み下す機能、言語により独立に読み速度を指定できる機能などを新たに追加した。

次節ではバイリンガル音声化システムの実現を中心に、BEP の構成と動作について述べる。

3 BEP の構成と動作

図 1 に示すように、BEP は Emacs Lisp 部とスピーチサーバ部からなる。

Emacs Lisp 部は OS によらず Emacs が動作するプラットフォームで共通に利用可能である。この Lisp 部では、(1) Emacs Lisp の hook や advice 機能を用いて、Emacs の動作を邪魔することなく Emacs の動作に即して音声出力すべき情報を収集し、(2) 取り出した情報を聴覚に適した形に変換し、(3) 声質情報などとともにスピーチサーバに送り出す。日英の言語読み分けに関する処理もここで行なう。スピーチサーバは実際に音声合成ソフトウエア（TTS）を制御するネイティブプログラムである。Linux 版では日本語と英語にそれぞれイン

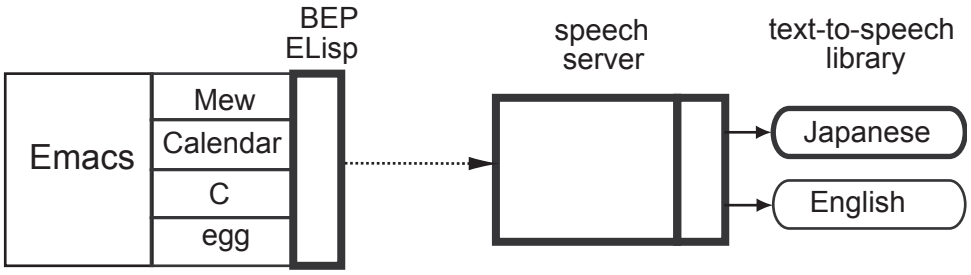

図 1: Linux 版 BEP の構成。太線枠部が今回開発した部分。

ターフェースの全くことなる TTS を用いるため、共通の TTSAPI を定めて制御を行なっている。Windows 版スピーチサーバでは Microsoft Speech API を用いることで統一的な制御を行なう。以下それぞれの詳細について述べる。

3.1 Lisp 部

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。フロントエンド部には Emacs 用の各種プログラムを音声環境で使いやすくするためのパッケージ群も含まれる。

3.1.1 バイリンガル拡張の仕組み

初期の BEP は日本語を用いる環境でも利用できることが主目的であったため、日本語対応機能の実装は Emacspeak 自体を書き換え、その機能を変更することで実現していた。たとえば、Emacspeak では 8 ビットめが on の文字（コードが 0x80 以上の文字）はそのまま扱わず、“octal NNN” のように変換して発声する他、これらの文字のみからなる行は無視されるため、この制限を緩和しなければならない。また、日本語には大文字・小文字の概念がない反面、ひらがな、カタカナ、漢字を識別して説明読みなどを行なわなければならないので、指定した文字 1 文字を読み上げる機能も変更する必要が生じる。初期リリースではこういった制限に対する変更を Emacspeak ソースを直接変更することで加えていた。

しかし、このような方法では本家 Emacspeak に機能が追加されるたびにその変更を一つ一つ BEP に取り込まなければならない。また、日本語以外の言語を意識していないため、さらなる拡張を考えると有効な手段とはいえない。

今回の新しい実装 [5] では、Emacspeak の構成ファイルを直接変更することは避け、まず、コア部分に属するいくつかの関数を多言語対応に拡張したもので再定義する。その上で、存在しえる多言語拡張機能の一つとして日本語拡張機能を組み込む構造とした。たとえば、Emacspeak において指定した 1 文字をフォネティック読み（例；`A` → `Cap Alpha`）する関数が呼び出されると、BEP で再定義した関数によってまずその文字の言語属性（後述）がチェックされる。そしてそれが英語（en）ならオリジナルの Emacspeak と同じ処理が呼び出され、日本語（ja）なら関数名の末尾に “`-ja`” をつけたものが呼び出される。そしてその関数はひらがな、カタカナ、漢字を識別し、漢字の説明読みを行なう。同様に、同じ 1 文字 “(” でも日本語属性を持っていれば「かっこ」と読み、英語なら「left paren」と読み上げる。

3.1.2 言語属性の付与

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。BEP のバイリンガル拡張は、この voice-lock にヒントを得た。

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。BEP における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、後述するように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。

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からである。そのため BEP 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。

3.1.3 三つのバイリンガルモード

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付与のための機能が呼び出されるが、割り当てる言語属性の判定規則を以下の 3 種から切り替え可能とした。このモードはユーザが任意の時点で切り替えることもできる。またバッファローカル変数であるので、たとえばモードごとに自動的に切替えることもできる。また、後から追加することも可能である。現在プリセットとして提供している三つの方式を以下に示す。

- 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。
- 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文字列の一部と判定し、日本語の音声合成に送ってカタカナ読みさせる。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。
- 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。

3.1.4 パッケージ対応の強化

BEP ではオリジナルの Emacspeak が対応している Lisp プログラム群に加えて、日本でポピュラーな下記のようなプログラムへの対応を強化している。

メール/News: Mew, Wanderlust 記事リストを表示する summary バッファでは削除や処理マークのある記事を音声を変えることでわかりやすく提示する。また、記事中で引用符によって引用されている部分は棒読みして識別可能とした。その他にも、各種イベントの音声または auditory-icon による通知、差出人など特定の情報を簡単に知ることができるキーコマンドの追加などを行なった。

WWW: emacs-w3m emacs-w3m では face（フォント属性）に応じて声質を変えることでアンカーや見出しをわかりやすく提示するようにした。

文書作成: YaTeX, YaHTML LaTeX ソース, HTML 作成でコマンドやタグを通常と異なった声質で提示する他、自動挿入などのイベントに応じて必要な部分を読み上げるようになった。

3.2 バイリンガルスピーチサーバ

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。

3.2.1 Linux 用スピーチサーバ

Linux 用スピーチサーバ（以下 Linux SS）は初期リリースの後で再設計し、日英のバイリンガルを実現するとともに、安定性と拡張性を高めた。

Linux SS 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。

3.2.2 共通 TTS インターフェース

Linux においては今のところ、標準化された音声出力のためのインターフェースがない。TTS ソフトウェアベンダーはそれぞれにプログラミングモデルの異なる独自の API を定めており、それに沿ってプログラムを開発することが求められている状況である[9]。

Linux SS で利用する TTS ソフトウェアも日本語と英語で全く異なった 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である。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 ViaVoice TTS 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コマンド文字列を挿入することで読み上げ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、それに合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。

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
- （言語によって速度を変えるための）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

[9] GNOME デスクトップ環境で音声入出力を扱う GNOME-Speech という枠組もあるが、開発途上であり、GNOME に特化しているため利用できない。

- 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、サウンドデバイスを解放してリクエスト待ち状態に移行する。

3.2.3 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おり、それぞれコマンドによって設定できる。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。

  コントローラ部で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、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。

3.2.4 Windows 用スピーチサーバ

Windows 用スピーチサーバでは Linux 版と同様のコマンドセットを入力とし、Microsoft Speech API 準拠の TTS ソフトウェアを制御して音声出力を行なう[10]。

初期のリリースにおいて、Windows 用スピーチサーバではキューに入力される文字列中に日本語（Shift_JIS）文字が存在するかどうかで日本語と英語の読み分けを行なっていた。今回の改良によって BEP は Lisp 部で言語の判定を行なうようになったので、Lisp 部から入力される言語指定コマンドに即して利用する TTS ソフトウェアを切替えるように変更した。また、Linux SS で採用している英語のカタカナ読み、ローマ字読み下しの機能[11]を採り入れた。

4 BEP の現状と今後

4.1 BEP の公開と利用

BEP は 2001 年 1 月に初版を BEP の Web サイト[12]で公開し、視覚障害者関連のメーリングリストで告知を行なった。それがきっかけとなって、Windows を中心に視覚障害を持つパワーユーザに利用されるようになった。その後、開発者とユーザを含む BEP メーリングリストで意見を求めつつ開発を行なっている。

Linux 版の配布は、BEP のスピーチサーバと Lisp パッケージからなり、ユーザは商用の TTS ソフトウェアを配布パッケージとは別に入手して独力でインストールする必要がある。Windows 版は、BEP を Meadow[13]に組み込んだ形のインストーラを作成して配布しており、TTS ソフトウェアを所有しているユーザはインストーラを実行するだけで手軽に BEP を体験できるようになっている。上記インストールに必要なものはすべて BEP の Web ページから最新版を入手可能である。

4.2 ユーザによる評価

現在 BEP を日常的に使用している視覚障害者は 10 名程度と考えられる（BEP メーリングリストでの発言者ベース）。ユーザは日頃から一般の視覚障害者用音声

[10] 英語には Microsoft の TTS エンジン、日本語には IBM Protalker97 または東芝 LaLa Voice 付属の東芝音声合成エンジンを利用する。

[11] 一般に日本語音声合成はローマ字を正しく読むのが苦手である。そこで BEP では、スピーチサーバ内部で正しい読み方に変換している。

[12] http://www.argv.org/bep/

[13] Meadow は、GNU Emacs の Windows 対応機能強化版。

化アプリケーションを Windows または MS-DOS 上で使いこなしている人がほとんどである。本プロジェクトでは BEP に関してユーザがどのように感じているかについて 2001 年夏頃にアンケート形式で意見を募集した。その結果のうち、主なものを以下に示す。

- よい点: バイリンガルである。他の Windows アプリに比べてキー操作で軽快に操作できる。音声種別によって読み上げる対象の要素を識別できる。メーリングリストを通じて開発状況がよく見え、サポートも受けられる。
- 欠点および改善の要望: 文章を斜め読みする機能がない。バッファ内の連続した読み上げにおいて中断したとき、中断位置に素早く飛べない。Emacs そのものであるため、メニュー操作に慣れた Windows ユーザにとって操作に馴染みにくい。インストールが難しい。動作が不安定である。日本語のドキュメントが少ない。

4.3 BEP の今後

上記に対処して、Windows 用のインストーラパッケージ作成、FAQ の作成などを行なったが、根本的な馴染みにくさは解決しきれていない。これは Emacs のキー操作が馴染みにくいことも一因であるが、それを単に Windows 標準に近づけるだけでは BEP（Emacs）の利点であるキーボードによる軽快な操作性を活用できない。今後はこれまで以上に、Windows ユーザを意識した入門ドキュメントの整備などが必要と考えられる。

上記のようなユーザの評価も考慮して、今後は以下のことに取り組みたいと考えている。

- 現在まだ不安定なバイリンガル拡張機能を Lisp 部、スピーチサーバ部ともに安定させる。
- 他の音声化エディタが備えているように、連続した読み上げに応じたカーソル位置の追従を可能にしたい。この機能を実装するには、スピーチサーバとの双方向の通信が必要となる。
- Linux 用のインストールパッケージ（DEB, RPM 等）を整備する。
- 日本語ドキュメントと Web ページを整備する。また、しばしば日本語以外の言語に関する問い合わせを受けるため、英語の Web や技術情報の整備も行なう。

その他の課題として、BEP 上で実用的に利用可能な音声 Web ブラウザの整備も必要である。Emacspeak は Emacs-W3 を CSS2 の音声スタイルシートに対応したブラウザとして利用できる機能を備えているが、W3 は動作が遅く日本語環境では問題も多い上開発が事実上停止している。Emacs-w3m は新たな選択肢になり得るが、本来 w3m は速度と画面表示を重視したブラウザであり、現状では音声ブラウザとして利用することは最適ではない。Emacspeak の開発者である Raman は XSLT を用いて Web コンテンツを音声で聞きやすい形に変換して提示する仕組みを導入しており、このようなアプローチも有効かも知れない。

BEP の新しい実装では、バッファ中の文字列をどの言語で読むべきかを確定する仕組みとその言語に対応したスピーチサーバさえあれば日本語と英語以外にも対応可能な設計となっている。日本と同じようなマルチバイトの文字セットを用いる言語（中国語、韓国語など）ではその言語と英語とを文字セットから判定することが比較的容易である。またこういった言語用の音声合成も開発されつつある。BEP が日本以外での視覚障害者のパソコン利用環境向上に利用できるかも知れない。

現在 Linux 版 BEP で用いている音声合成ソフトウェアは日英ともに商用のもので、入手やサポートが頒布元の意向に制限される。英語ではフリーの TTS ソフトウェアも登場している[14]が、日本語対応のものはまだ存在していない。BEP のような視覚障害者用音声化システムに用いるためには、高速な読み上げ、声のバラエティ、読み上げ開始/停止時の高いパフォーマンスが要求される。このような需要を満たすフリーの日本語 TTS の開発を呼びかけ、協力していくことも必要である。

また、音声だけでなく点字出力への対応も求められている。プログラミングやレイアウトを重視する文書など、音声による時系列的情報提示では十分な作業効率が得られない場合が多いためである。Emacs 内部から取得したカーソル周囲の情報、イベントの通知を点字に変換し、点字を表示するハードウェア（点字ピンディスプレイ）に出力することになる。日本語文字列を点字に変換するフリーのソフトウェア、点字ピンディスプレイの制御モジュールなどが必要となるため、Emacspeak とは直行する独立したソフトウェアで実現することになる。

[14] エジンバラ大学の Festival を元とした FLite が知られている。

5　まとめ

著者らは Linux および Windows に対応した日英 2ヶ国語の GNU Emacs 音声化システム BEP を開発している。BEP は 3 種類のバイリンガルモードを持ち，日本人に適した英語の音声化ができる。

国内では Linux をコンソールから直接音声利用できるソフトウェアで一般に配布されたものがまだなく、BEP は Linux を日本語環境で実用的に利用可能な唯一の音声化システムである。BEP の利用には Emacs への習熟が必要であることから現在の利用者はそれほど多くないが、今後 Linux の利用が一般化するにしたがって需要は増えると考えられる。

BEP は GNU GPL の元でオープンソースとして開発しており、商用の TTS ソフトウエアを除くソースコードおよび詳しい情報を BEP の Web サイトから入手できる。開発は BEP メーリングリスト上での議論を元に行なわれており、誰でも参加することができる。今後、多言語対応の安定化、実用的に利用するために希望の多い機能の実装、他の言語へのサポート、さらには PDA への対応、点字出力を行なう新たな枠組の開発などを検討している。**慢性的に開発者が不足しているので、興味を持たれた方は是非御協力いただきたい。**

謝辞

GNU やオープンソースという言葉によって代表される知的財産の共有文化に感謝する。坂本貢氏、高橋玲子氏、南谷和範氏、吉本浩二氏、中村典嗣氏、そして高橋直人氏など、ユーザ及び開発者として貢献し有益な助言を授けてくれた BEP のメンバー達に感謝する。また、オープンソースとして或いは無償で開発成果を利用させていただいた T.V. Raman 博士、G. Bishop 教授、「グラスルーツ」の開発者である石川准教授に感謝する。本研究の一部は、日本学術振興会未来開拓学術研究推進事業における研究プロジェクト「高度マルチメディア応用システム構築のための先進的ネットワークアーキテクチャの研究」（JSPS-RFTF97R16301）による。また一部は情報処理振興事業協会「平成 13 年度未踏ソフトウェア創造事業」による。ここに記して謝意を表す。

参考文献

[1] 石川 准: “GUI 用スクリーンリーダの現状と課題 – 北米と欧州の取り組みを中心に –”, 情報処理, Vol. 36, No. 12, (1995) pp. 1133〜1139.

[2] T.V. Raman: “Auditory User Interfaces – Toward the Speaking Computer–”, Kluwer Academic Publishers (1997).

[3] 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).

[4] 渡辺隆行、井上浩一、切明政憲: “視覚障害者用日英 2ヶ国語音声出力システム BEP”, 電子情報通信学会 思考と言語研究会 (2002 年 7 月 29 日、ATR)、信学技報 102/254，pp.15–22.

[5] 渡辺隆行、切明政憲: “平成 13 年度 IPA 未踏ソフトウェア成果報告書”, http://www.argv.org/bep/doc/mito13report/FinalReport.html.

付録

A　英語音声化に 2 種類の発音が必要な理由

BEP のユーザ 9 名に、BEP を使う際にカタカナ英語（以下、K 英語と略す）とネイティブ英語（N 英語）をどう使い分けたいかインタビューした。9 名の内訳は男性 8 名と女性 1 名、全盲 5 名と弱視 1 名と晴眼 3 名である。英語の聞き取り能力に関してはほぼ不自由なく聞き取れるユーザから英語が苦手なユーザまでさまざまである。インタビューの結果、以下のことが明らかになった。

- 英語の聞き取り能力が高いユーザほど N 英語を好む。
- N 英語には、イントネーションやアクセントが自然であるという利点と、K 英語では区別できない ‘right’ と ‘light’、‘bat’ と ‘but’ のような子音や母音を区別できるという利点がある。

- K 英語を使うと、もともとカタカナで書かれていたのか、それとも音声化システムがもともとの英語を K 英語に変換したのかわからなくなる。N 英語の場合、その情報が英語で書かれていることがわかる。
- 英語で書かれている情報は N 英語で聞きたいという要求がある。したがって、N 英語で発音する機能が絶対に必要である。
- Web を読むときなどに、日本語と N 英語を自動的に切り替えて音声化するシステムを必要としているユーザがいる。
- N 英語と K 英語のどちらを使いたいかは、それぞれの合成音声の品質によっても変化する。つまり、聞きやすい声であることの方が重要である。
- どちらに慣れているかにも影響される。
- 一般に日本人の英語聞き取り能力は日本語聞き取り能力より劣るので、N 英語で読み上げる場合、日本語よりも速度を遅くした方がよい。
- 十分な英語の聞き取り能力を持っていない場合、いきなり N 英語で読み上げられると何を言っているのか聞き取れない。したがって、短い単位で日本語と英語が切り替わる場合は K 英語で発音した方がよい。その意味でメニューやシステムメッセージなどは K 英語の方がよい。
- 情報の長さが長いほど K 英語より N 英語が適するようになる。
- 日本人ならば K 英語で読み上げても聞き取れる。
- ある程度英語で議論できる程度の聞き取り能力を持ったユーザでも、仕事で読まなければならない科学技術文献は、正確に理解するために K 英語か点字を使いたいという要求がある。
- 逆に文系の文書は N 英語の方が適する。

B リンク集

BEP 関連

・BEP の Web サイト;
http://www.argv.org/bep/
・BEP のビデオおよび音声によるデモ;
http://www.sfc.keio.ac.jp/ wata7be/bep_demo/BEPLinux/
・クリエートシステム開発のドキュメントトーカ;
http://www.createsystem.co.jp/
・UNIX 対応のフリーな音声合成 Festival;
http://www.cstr.ed.ac.uk/projects/festival.html
・フリーで計量な音声合成 FLite;
http://cmuflite.org/

Emacspeak 関連 (英語)

・GNU Emacs;
http://www.gnu.org/software/emacs/
・Emacspeak 公開サイト (sourceforge);
http://emacspeak.sourceforge.net/
・T.V. Raman の Web サイト;
http://cs.cornell.edu/home/raman/
・Introduction of Emacspeak;
http://leb.net/blinux/emacspeak_intro.html
・Voices about Emacspeak;
http://leb.net/blinux/emacspeak_voices.html

スクリーンリーダ

・UNIX 用英語スクリーンリーダ SPEAKUP;
http://www.linux-speakup.org/
・UNIX 用英語点字出力デーモン BRLTTY;
http://mielke.cc/brltty/
・GNOME Accessibility Project;
http://developer.gnome.org/projects/gap
・Linux Accessibility Resource Site;
http://trace.wisc.edu/linux/

アクセシビリティ関連の Web サイト

・W3C WAI (Web Accessibility Initiative);
http://www.w3.org/WAI/
・CSS2 (Cascading Style Sheets, level2);
http://www.w3.org/TR/REC-CSS2/

BLINUX

・BLINUX (Blind+Linux) の Web サイト;
http://leb.net/blinux/
・Blind Unix Users' Group(BUG): 日本語;
http://www.argv.org/ inoue/bug-ml.html