第 8 部

マルチメディア (Phone Shell)

第 1 章

phone shell の設計とネットワーク管理への応用

1.1 はじめに

現在, 我々が日常的に接するさまざまなシステムは大規模化が進行し, これに伴ってシステム側の世界と, 人間が活動する実世界とのインタフェースも複雑・多様化し, この対策が重要な課題となっている. そのため, 従来のアプローチを見直し, 音声や画像を含めた新たなメディアの利用を考慮する必要が生じている.

ネットワーク化された大規模分散システムも例外ではない. すなわち, 実世界側に要求されているシステム全体を把握する業務と, それに連携するコンピュータ環境側の技術であるネットワークマネージメントとの関連を見直す段階にさしかかっている.

システムやネットワークの管理や運用といった業務は, 関連技術の発展によって自動化や簡略化が進行しているが, 業務自体の知的内容や故障対応時間などに対する要求は高度になる一方である. 例えば, 24 時間連続稼働する大規模なネットワークシステムにおいては, システム全体をくまなく把握し管理している少数の人間（システム管理者）とシステムとの情報伝達が, 管理者の位置や時間によって制約されてはならないが, この要求に対応するには, 実世界側の管理者と, コンピュータとのインタフェースの形態を大幅に改善する必要がある.

このような状況下では, システムが音声情報を取り扱うことができ, さらに電話を活用して, 音声を用いたさまざまなコミュニケーションを, 管理者とシステムとの間に随時確立する能力があれば, この問題を解決する有力な手段となる. これは, 従来のディスプレイとプリンタにのみ依存するインタフェースに新たに加わる有効なアプローチである. なお, この目的のために導入される音声情報を活用するシステムは, システム管理・運用のみならず, システムの利用者とシステムとのより有効なインタフェースとして広く一般的に利用することもできる.

本論文で報告する phone shell は, 広域・大規模分散環境に基づいた次世代のコンピュータ環境構築技術の研究開発を目的とする, WIDE プロジェクトにおける一連の音声関連の研究の成果の一つであり, システムの管理運用方法の改善を目指した, WIDE プロジェクトからの提案である. この, phone shell は, 電話のタッチトーン（DTMF 信号）を入力とし, 計算機によって合成された音声を出力とする一種の shell で, プッシュホンがあればどこからでも利用することが可能である. phone shell のこの特徴は, 管理者の位置や時間

によって生じるさまざまな制約を緩和する. 事実, 既に WIDE インターネットの環境で実際に稼働している phone shell は, ネットワークの管理運用に成果をあげている.

1.2 phone shell の構成

phone shell は, プッシュホンを介して UNIX ワークステーションを操作するために考案されたシステムである.

このシステムが想定している標準的な利用者は, UNIX ワークステーションの操作に精通した大規模分散環境の管理者である. システム管理者は常にシステムの稼働状況に気を配り, 必要に応じて的確な措置を迅速に実行することが強く要求されており, 管理者にとっては, この問題にいかに対応するかは, 常に重要かつ困難な課題のひとつとなっている. phone shell は, この問題に対する一つの試みとして, システム管理者に, 公衆電話(プッシュホン)からネットワーク管理を実行する手段を提供するものである.

phone shell は, その利用者にはプッシュホンから送出される DTMF 信号を標準入力に取り込み, 音声を出力する変則的な shell に見えるが, 実態は以下に述べるハードウェアとソフトウェアから構成される, UNIX のコマンドプロセッサ (shell) のフロントエンドシステムである.

1.2.1 phone shell のハードウェア

phone shell を構成するハードウェアの概略を, 図 1.1に示す. 図から明らかなように, phone shell は,

- 音声合成系
- 電話回線制御系

のふたつに大別される.

音声合成系

phone shell が利用している音声は, 全て人間の声を単語ごとに別々に録音したものである. 現在, 日本語の男声と英語の女声の 2 種類をそれぞれ約 350 語用意している. どちらも,

- host 名, login 名のような固有名詞
- 数詞
- アルファベット
- “Good morning”, “おはようございます” といった特定のフレーズ

から構成され, 日本語の場合にはこれにひらがな（五十音）が追加されている. なお, これらの音声は全て サンプリング周波数 38KHz, データ幅 8bit, データフォーマット μ-Law で記録されている.

ところで, 最近のワークステーションや X ウィンドウ端末には, 音声合成機能を搭載した機種が多い.1 機種で何種類かの音声データフォーマットを受け付ける機種もあるが, 全ての機種に共通するデータフォーマットはない, また広く普及した標準的なデータフォーマットといったものもない.

現在 phone shell が利用する音声データは全て上記の形式で記録されているが, 将来 2 箇所以上の異る機種上で phone shell を稼働するようになった場合, 音声データが共有可能であることは重要である. また, phone shell 以外の目的に音声情報を利用することもあるので, その点からも 音声情報を共有できるようあらかじめ準備しておくことが望ましい.

そこで, 現時点では上記のフォーマットを WIDE プロジェクトにおける標準的な音声データフォーマットとし, その他のフォーマットとの相互変換のためのプログラムを用意することにした.

変換プログラムの名称（フルパス名）やそのオプションは, /etc/printcap を参考にして新規作成した/etc/soundcap なるファイルに記載し, 同じく新規作成した音声制御用のデーモン/etc/soundd が音声データの変換を行なう際に参照するようにした. /etc/soundd は, printer 環境における /etc/lpd のような振舞いをするデーモンである. なお詳細は省略する.

本方式では, 音声情報の共有化とは A/D 変換の形式やサンプリング周波数といったレベルでの相互変換の実現を指している. これは極めてプリミティブなレベルでの共有化である. これに対して音声情報の共有は, その意味や音韻に基づいたもっと抽象度の高いレベルで行なうべきであり, 本方式のような低レベルでの互換を実現しても意味がないという主張もある. これは極めて妥当な主張であるが, 現在音声情報は単なるバイナリデータとしてしか取り扱う方法が存在せず, 抽象的な操作が困難であるとの理由から採用を見送った. しかし今後も音声情報の共有方法については検討を続けることにする.

電話回線制御系

電話回線を制御する部分には, 市販の NCU を用いた. この NCU は, 通常のモデムが持つ回線制御機能を一通り持ち, さらに通話状態においては相手側から送られてくる DTMF 信号を解読することができる. なお, 通常プッシュホンは, 12 個ボタンで構成されているが DTMF 信号は図 1.2 のように合計 16 個割り当てられており, 本 NCU も, 通常の 12 個の他に A から D までの信号（以後, 拡張トーン）を認識できる.

1.2.2　phone shell のソフトウェア

phone shell は, 上記の NCU を制御し, DTMF 信号をあらかじめ決めたルールに沿ってに ASCII 文字に変換し, ユーザの指定する shell の stdin に送る. 同時に, アプリケーショ

ンやカーネルが stdout や stderr に出すメッセージを観察し, phone shell の実態である phsh が保持するテーブルと照合して, 発声する必要がありかつ可能であればこれを読み上げる.

今回は実装や改良の容易さから, NCU の制御と音声合成装置の制御は pncd, sounddの 2 つのデーモンに担当させ, これらのデーモンと shell とのやりとりは phsh が担当するようにした. なお, phsh は pty を利用している. これによりカーネルには全く手を付けることなく phone shell を作成できた.

1.2.3 phone shell の keybindings

プッシュホンの 12 個のキー（図 1.3）を入力装置として利用するアプローチは以前からなされており, 飛行機や列車の座席の予約サービスにおいては既に実用化されている. また, 数字・ひらがな・あらかじめ決められた文面・イラストなどを送ることが可能なディスプレイポケットベルのような製品も普及を始め, さらに, 障害のために発声ができない人が電話を使って意志を伝達するための手段として利用できないか, といった検討もなされている [80] .

しかし, これらの例では, 入力可能な文字は数字のみあるいは数字とひらがな程度である. これに対して, phone shell では UNIX を操作するために, 英数字に加えて記号もサポートしなければならず, 操作方法やキーバインディングについて十分な検討が必要となる.

ここでは, 以下の 3 つの方法を検討する.

- コード入力方式
- 縮退キー方式
- 多段シフトキー方式

コード入力方式

入力したい文字の ASCII コードを 8 進数や 16 進数で入力する方法である. 実現は容易であるが, ユーザに文字コードを全て暗記するか, コード表を見ながら操作することを要求するため, ユーザにかかる負担が大きい.

通常のプッシュホンではキーは 12 個しかないので, 16 進数方式を採用すると, シフトキーを用いるなどの対応が必要となり, 必要なキーストローク数が, 8 進コード入力に比べて多くなってしまう場合もあるので, 両者の優劣は一概には決められない.

なお, キーは全部で 12 あるので, 12 進数方式を採用すると 2 ストロークで 144 種類の文字を入力でき, ASCII 文字を入力するだけなら十分である. 利用者がコード表を覚えてしまえば効率良い入力が可能になると予想される.

縮退キー方式

欧米のプッシュホンタイプの公衆電話には, 数字キーに 3 文字ずつアルファベットが割り当てられている. 例えば 2 のキーには, A, B, C の 3 文字が割り当てられている. このような状況では, UNIX のコマンド cc, bc, cb はいずれも 2 2 となってしまい区別がつかない. これを「縮退キー」と呼ぶことにする. なお, 通常は, Q と Z はどのキーにも割り当てられていないが, 文字の割当のない 1 にマップすることにする.

ところで, 入力しようとする単語がコマンド名であるとかファイル名であるとかが事前にわかっていれば, 全ての文字を入力しなくても入力途中の単語の所属する名前空間を検索することで文字列を決定できる場合がある. 逆に, 縮退キーの場合には上記の例のように全ての文字を入力してもなお文字列を決定できない場合もある.

そこで, 縮退キーを用いるとどの程度の混乱が生じるかを調べた. 調査は, 著者のひとりが日常利用しているワークステーション上で一般利用者に公開されているコマンドおよそ 600 個を対象とし, コマンドを縮退キーで入力したとき, 何ストローク入力すれば他のコマンドと区別できるかを調べた. また, 比較のために通常のタイプライタ型キーボードで入力した場合も調べた. なお, 上記の cc, bc, cb の場合は, タイプライタ型では 2 ストローク目でコマンドが確定するが, 縮退キーではそれでも確定せず, 何らかの方法で 3 つのうちのいずれかを選択しなければならない. このような場合はそのためにもう 1 ストローク必要と考え 3 ストロークとして計算した.

その結果, コマンドを縮退キーで全て打鍵した場合の平均値は 5.7 ストロークであり, また, 平均 4.2 ストロークの打鍵でコマンドが確定することがわかった. また, およそ 5 % のコマンドが, cc, bc, cb の例のように同一の文字列に射影されていることが判明した.

一方通常のキーボードで同一の操作を行なうと, それぞれ 5.5 ストローク, 3.54 ストロークであり, 縮退キーであっても, 極端に打鍵数が増えるわけではないことがわかった.

この方式では, 記号の扱いが問題になる.UNIX では !, &, |, >, < などの 1 文字の記号が重要な意味をもつが, 上の統計では全て 0 に対応させて処理した. これは実用的な対応ではない. 実際には, 記号も 0 から 9 までのキーに分散して割り当てるか, コード入力方式とうまく組み合わせるといった修正が必要となる.

多段シフトキー方式

縮退キー方式が, 一つの数字キーに割り当てられた複数の文字を区別できないのに対し, この方式はシフトを指示するシーケンスを定め, それにより数字キーに割り当てられた複数の文字を区別する方式である.

この方式は, キートップに文字を刻印するなどの対策をとれば, 入力しやすい方法であるが, シフトが頻発し, キーストローク数が多くなるという欠点がある.

1.3 WIDE/VOICE service

phone shell の動作確認試験を兼ねて, ネットワークの稼働状況を点検することを主体とするテレホンサービスを試作した. このテレホンサービスは

- 開始時にパスワードなどを要求してこない.
- 電話番号と操作方法を知るものは誰でも利用できる.
- あらかじめ用意された作業しか実行できない.

などの特徴から, phone shell の “疑似ユーザ” と位置付けることができる. なお, このサービスは, WIDE/VOICE service と呼ばれている.

以下に, WIDE/VOICE service の操作例を 2 つ示す（図 1.4, 図 1.5）.

ここに示したのは, いずれも “version 0.5” と呼ばれる版のサービス例である. これは, あらかじめ用意された 10 種類のサービスのうちの 1 つを選択実行させ返答を待つ, という操作が基本で, この操作を任意回繰り返し実行できるようになっている. このサービスの実行を C 言語風に記述すると図 1.6 のようになる.

“version 0.5” には, 大別して 2 つのタイプのサービスがある. 一つは, 実行すると結果を音声で返してくるタイプで, 図 1.5 がその例である. もう一つは, 図 1.4 の例のように実行するとオプションを指定するように指示があり, それに答えると結果を音声で返してくるものである.

図 1.4では, 利用者は /usr/etc/ping に相当するサービスを行なう 3 を選択しており, コマンド選択後, 計算機から IP アドレスを入力するよう指示されている. 順を追ってゆくと以下のようになる.

1. 開始メッセージに続き, コンピュータは, 数字キーを一つ押すように指示.
2. 利用者は 3 を押下. 3 には, ping コマンドが設定されている.
3. コンピュータは IP アドレスの入力を要求.
4. 利用者は IP アドレスを入力. この例では 128.32.130.3 (okeeffe.berkeley.edu) を ping している.
5. コンピュータから “okeeffe.berkeley.edu is alive” との返答があった.
6. 利用者は終了を示す # を押下.
7. コンピュータは, 終了メッセージを送出し, サービス終了.（電話回線切断）

図 1.5では, 利用者はまず 0 を次に 1 を選択し, WIDE/VOICE service を運用しているマシンのロードアベレージと利用者数の報告を受けている.

WIDE/VOICE service “version 0.5” では, システム管理者が事前に用意したサービスしか実行できないので, サービスの質はシステム管理者がどのようなサービスを設定する

かに委ねられてしまうという点に問題があるが, 操作が簡単であるという点は評価されるべきである.

なお, WIDE/VOICE service は, phone shell の最初の版が動き始めた 1990 年始めから継続的に運用されており, ゲートウェイマシンの稼働状況などを常に把握できることから, 良好な評価を受けている.

1.4 phone shell の問題点

phone shell は, プッシュホンの 12 個のキーを入力装置とする, いままでにないタイプの shell であるため, その操作法の習得にどれほどの時間を必要とするか, あるいはもっと効率の良い手法はないかという疑問がある. そこで本節では, 音声認識装置の採用の是非と, プッシュホンのキーの操作の習得方法について検討する.

1.4.1 音声認識装置の利用

phone shell では プッシュホンの DTMF 信号を利用することを前提としているが, 音声認識装置を入力装置として利用することも当然考えられる.

たしかに, 人間と同レベルの認識能力を持つ音声認識装置が普及すれば非常に強力な入力装置となるので, 事実多方面から研究が行なわれている [81] [82].

音声認識装置には, 以下のような利点がある.

1. 操作が容易.
2. 利用者が本人であるか否かの認証が可能.

しかし現状では,

1. 騒音の大きな環境下で利用した場合に十分な認識率を維持することができない.
2. 声に出して指示を行なうため秘匿性がない.
3. 計算機に話しかけるという習慣がまだ一般的ではないため, 利用者が違和感を持つ.

といった問題がある. (2) について補足すると, 音声認識を用いれば音声によりユーザの認証が可能なので, login 時のパスワードの照合などは声を出さずに済む. また, もし必要ならパスワードだけはキー入力で行なうことも考えられる. したがって, この点に関しては音声認識方式を採用しても問題はない. しかし, 音声で操作を行なうと操作内容が全て周囲に聞こえてしまう点が問題である.

いずれにせよ, 音声認識装置をワークステーションの入力装置として利用し, 音声による指示で大規模分散環境の管理を行なうには, 少なくとも以下の条件が満たされる必要がある.

1. 複数の話者の音声を識別可能であること

2. 認識可能な単語の数が最低数百あるいはそれ以上のオーダであること.

3. 認識率が十分高く, 誤って認識された場合の訂正が容易であること.

4. 周囲の騒音などの影響を受けないこと.

5. 電話回線を経由した音声（3.4kHz 帯域のフィルタを通過した音声）でも認識が行えること

上記の条件 (1), (2), (3) を満たす, 実用的なレベルに達した製品はいくつもあるが, (5) の条件を満たすことが必須である場合には, 十分な認識率が得られる音声認識装置は, まだ研究段階である. [83]. さらに, 音声認識装置のみでは, ファイル名やホスト名など固有名詞を効率良く入力するのは難しく, 声に出して入力するのは危険な情報も少なからずある.

結局, 音声認識をのみを入力装置として採用するのは現時点では難しく, キー入力との併用は有効かも知れないが, 絶大な効果が望めるわけではない, との判断から今回は採用を見送った.

1.4.2 操作法の習得

12 個のキーのみでワークステーションを操作する, phone shell は, 決して使いやすいものではない. しかし, ネットワークのトラブルは突発するものであり, プッシュホンだけで状況の把握や対策を講じることが可能になる意義は大きい. したがって, phone shell を利用する可能性のあるシステム管理者は, 事前に操作法を訓練しておくのがよい.

そこで, 2 つのプログラム, twintty と pipopa を用いる簡単なキー操作練習環境を用意した.

まず, twintty は, /usr/ucb/script のほぼ逆の発想のプログラムで, shell からの出力には何も手を加えず素通しとし, shell への入力に関しては, 通常のキーボードからの入力以外に, スクリプトファイル（あるいはソケット）からの入力も受けつけるというものである. 一方 pipopa は, 端末のキーボードの一部分あるいは, テンキーの部分をプッシュホンのキーと見たて, ユーザの入力する文字列（数字列）を, 2 節で述べた方式のうちのいずれかで順次 ASCII 文字に変換し, ファイルあるいはソケットに出力するプログラムである. twintty と pipopa を組み合わせれば, ユーザは端末から phone shell の操作感を習得することができる. （図 1.7）

ところで, 実際にプッシュホンを操作すると, キーを押すごとにそれに対応した DTMF 信号が利用者にフィードバックされる. このフィードバックの有無が, 操作の上達におよぼす影響は無視できないと思われるので, 後述する「試作版ダイアラ」の DTMF 信号発信器をワークステーションに接続し, twintty ＋ pipopa による練習環境でも DTMF 信号が利用者にフィードバックされるようにし, フィードバックの効果を検討できるようにした.

また, どのようなキーバインディングがより有効か, 上達にはどれくらいの練習を必要とするのか, などについて調べることは重要であり, そのためには pipopa を用いて phone shell の操作を習得している過程の打鍵状況を記録しておくことが望ましい. 現在, 筆者

のうちのひとりが, パーソナルコンピュータ用の打鍵情報記録システム [84] を用いて観察を行なっている.

なお, 片手で操作することが可能なキーボード [85] や, キーボードの操作を効果的に学習するための手法 [86] , などは以前より, 提案されており, キーボードの操作性を評価するための手法も検討されている [87] ので, phone shell の操作性の評価にあたっては, これらの成果を十分考慮する必要がある.

1.5　phone shell の利用環境の整備

最後に phone shell のより有効な活用を支援するための「道具」について検討する.

1.5.1　トーンダイアラの利用

繰り返し述べてきたように, phone shell を利用してワークステーションを操作するのに必要となるのはプッシュホンだけであり, モデムやパーソナルコンピュータといった装置を必要としない.

しかし, 残念ながら現時点では DTMF 信号を送出する機能のない電話器が家庭用, 公衆電話ともに未だ数多く利用されているため, phone shell の利用者は DTMF 信号発信器（トーンダイアラ）を携行せざるを得ない. 幸い, 安価で小型軽量のトーンダイアラが数多く市販されているので, これを持ち歩くことによって生じる負担は, モデムとパーソナルコンピュータを持ち運ぶことに比べれば, 無視できる規模である.

ところで, トーンダイアラによる DTMF 信号の送出は, その構造上の理由で周囲の雑音を取り込みやすい. そのため,

- 騒音の大きな環境でも DTMF 信号を正確に送出できるか否か

は重要で, トーンダイアラのスピーカの構造や音圧レベルの設定が適切である必要がある. phone shell の効率よい利用を実現するには, これに加えて以下のような機能が望まれる.

- IP アドレスなど, 利用することが事前にわかっているキーワードの数字列（例: 128 ＊ 32 ＊ 130 ＊ 3 ＃）をあらかじめ登録しておく機能.
- 登録されている数字列を DTMF 信号に変換して送出する機能.
- 外部の計算機との間でデータをやりとりする機能.

また, ユーザが利用しやすいよう, ユーザインタフェースに工夫が施されていることも望まれる.

もし, これらの機能をもつトーンダイアラがあれば, DTMF 機能をもつ電話から利用する場合であっても有効であろう. また, 何よりも phone shell をデバッグする際に威力を発揮する.

しかし, 残念ながらこれらの要求を全て満たす製品は今のところ見当たらない. そこで市販のトーンダイアラを改造した DTMF 信号発信器と, パーソナルコンピュータを組み合わせて試作した.

今回試作したトーンダイアラ（以後, 試作版ダイアラ）の構成を図 1.8に, DTMF 信号発信器の外観を付録（図 1.10）に示す.

DTMF 信号発信器は入力装置として RS-232C を持つだけなので, 単独では利用できず, ラップトップコンピュータなどをフロントエンドコントローラとして併用しなければならない. このため試作系を用いて phone shell を利用するのは, かなり大がかりなものになってしまった. しかし, 図 1.8において破線で示した範囲を, 現在のダイアラの大きさに収めることは技術的には全く問題ないので, ここでは形態についてはこれ以上議論しないこととする.

ところでトーンダイアラを用いて DTMF 信号を送出するには, トーンダイアラのスピーカを送話器に押しあてて操作を行なう必要があるが, 送話器の感度や周囲の騒音によっては, 送出した DTMF 信号が認識されない場合がある. ちなみにトーンダイアラを用いた入力では, 操作に対するフィードバックが DTMF 信号のみなので, 送出は確認できるが正しく認識されているか否かは確認する方法がない.

そこで, A〜D の拡張トーンを同期点 (synchronization point) として利用し, DTMF 信号のとりこぼしが生じた場合にそれを検出することを検討した.

例えば, 131 ＊ 112 ＊ 40 ＊ 1 ＃ と送出する場合に,

1	3	1	＊	1	1	2	＊	4	0	＊	1	＃

をとせず, 以下のようにする.

A	1	B	3	B	1	B	＊	B	1	B	1	B	2	B	＊	B	4	B	0	B	＊	B	1	B	＃	B	D

ここで, A は入力文字列の開始を D は入力文字列の終了を示し, B は各文字の終了を示すものとする. この場合, ストローク数が 2 倍以上になるので, 一文字ずつ人間が手で入力していると, 伝送エラーが検出される可能性より, 人間の入力ミスが生じる可能性の方が高くなってしまうことが予想されるが, 試作版ダイアラが拡張トーンを自動挿入するよう設定すれば問題はない.

拡張トーンの利用方法は他にもいくつか検討したが, 今回は上記の方法を用いて, phone shell を操作する実験を行なった. その結果, DTMF 信号が正確に伝わらず, 的確な操作ができないような騒音の多い環境下でも, DTMF 信号のとりこぼしだけは的確に検出でき, 再入力を促すなどの対応が可能で, 信頼性の向上に寄与することが確認できた.

なお, 前述のように, 現行の試作版ダイアラは, 気軽に持ち運んで利用する形態ではないが, 大きさの問題は技術的に解決可能であるので, この点にはこだわらず, 今後は試作版ダイアラにどのようなユーザインタフェースを搭載すれば利用者にとって便利であるかを検討することにしている.

1.5.2　ファクシミリとの連携

phone shell が採用した音声による情報提供機能は, 電話によるシステム管理への道を開いた点で有効であったが, やはり印刷された情報が必要となる場合もある.

たとえば, あるマシンに接続されているディスクの使用状況を知りたいという要求があったとする. この要求がたとえば, capacity が 99% を超えたファイルシステムの有無を知りたい, といったものであれば, 音声による情報の提供で十分に対応できる. しかし, ファイルシステムの使用状況を具体的に知りたいという要求に対して, 音声合成機能を用いて読み上げるのは可能ではあるが現実的ではなく, /bin/df コマンドの出力をそのまま提示するのがよい. このような要求に答えるため, ファクシミリとの連携を検討した.

ファクシミリ装置をワークステーションあるいはパーソナルコンピュータの出力装置として利用するには, いくつかのアプローチがあるが, 今回は中村 [88] の方法を検討した.

この方法は, 単なるテキストファイルではなく, dvi 形式のファイルをファクシミリ装置に送出するものなので, 単なるテキストファイルを送出する場合には, 簡単なフィルタを用意し図 1.9 のような処理が必要になる. この例では, foo というテキストファイルを JLaTeX で処理し, 出来上がった dvi ファイルを, ファクシミリにて 03–123–4567 に送付している.

この方法では, 数式や表を含むような文書を TeX で処理し, 印字品質を劣化させずに送出することも可能である. 今後, TeX に加えて, PostScript のファイルがファクシミリで送れるようになると, 応用はよりいっそう広がるであろう.

ファクシミリとの連携は, phone shell の応用範囲を広げ, 例えば, 出張途中の新幹線の中から phone shell を用いてシステムのチェックをし, 問題が見つかった場合には詳細を, 宿泊先のホテルにファクシミリで送付するといった作業が可能になる.

1.6　おわりに

大規模分散環境の管理運用の形態を改善するための一つのアプローチとして, 音声とプッシュホンを活用した phone shell を提案した. また, phone shell を活用するための道具としてトーンダイアラやファクシミリの併用を提案し, 音声認識装置の利用可能性も検討した.

今後は, まず phone shell のユーザインタフェースを改善し, 大規模分散環境管理用の実用的なツールの水準にまで引き上げるとともに, 音声情報の効率良い運用方法についても検討する.

付録

［試作したトーンダイアラ］

今回試作したトーンダイアラの外見を図 1.10 に示す. これは市販のトーンダイアラのキーパッドの部分を取り外し, その部分にインタフェースユニットを組み込んだもので

ある.

インタフェースユニット上には,

- レベル変換
- シリアルパラレル変換
- キーマトリクス制御

のための IC が配置されており, 外部から RS-232C を経由して文字コードが送られると, それに対応したキーが押されたものとして, DTMF 信号を発生する. もし, 0〜9, ＊, #, A, B, C, D 以外の文字コードが送られた場合には無視される.

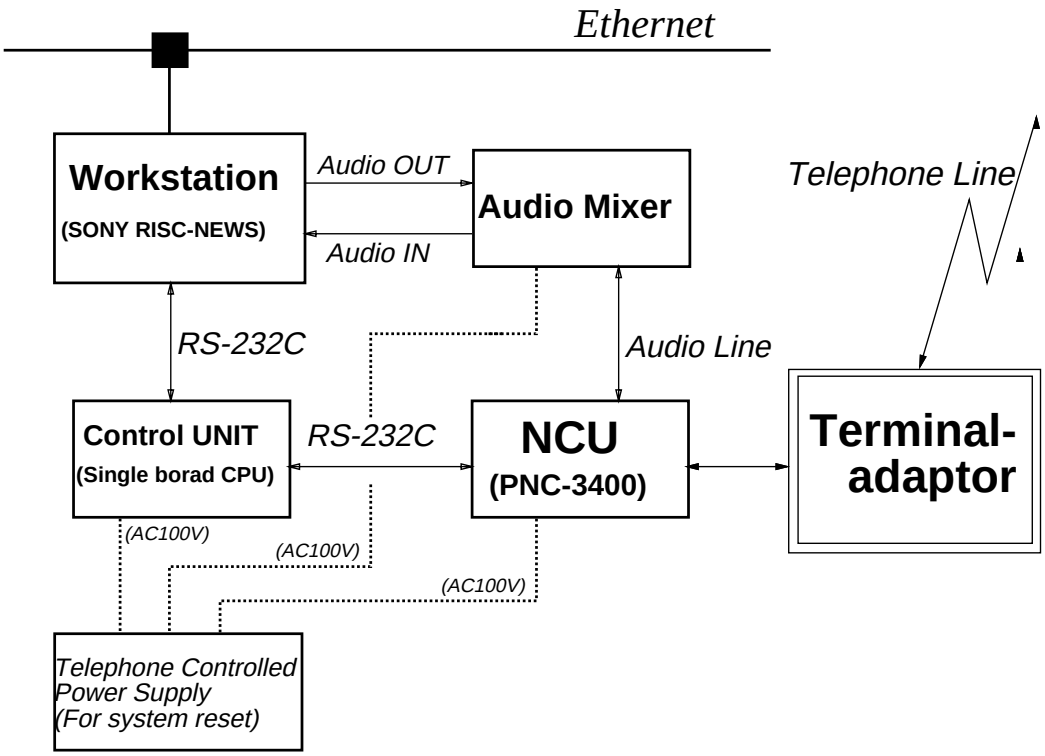

図 1.1: phone shell のハードウェア

Hz	1209	1336	1477	1633
697	1	2	3	A
770	4	5	6	B
853	7	8	9	C
941	＊	0	#	D

図 1.2: プッシュトーンの周波数のマトリクス

1 Q Z	2 A B C	3 D E F
4 G H I	5 K L M	6 M N O
7 P R S	8 T U V	9 W X Y
＊	0 OPER	#

図 1.3: プッシュホンのキーボード

computer: This is WIDE information service. Please push number key.

user: [3]

computer: Please enter the IP address.

user: [1] [2] [8] [＊] [3] [2] [＊] [1] [3] [0] [＊] [3] [#]

computer: Okeeffe dot berkeley dot edu is alive. Please push number key.

user: [#]

computer: This is the end of the information service. Thank you.

図 1.4: phone shell の操作例 (1)

:

computer: ... please push number key.

user: [0]

computer: Load average of dixon is zero point three five.

computer: Please push number key.

user: [1]

computer: Three users are working on dixon. Please push number key.

:

図 1.5: phone shell の操作例 (2)

```
1:  main(){
2:      char    key;
3:      opening_message();
4:      while((key = menu()) != '#') {
5:          service(key);
6:      }
7:      ending_message();
8:  }
```

図 1.6: WIDE/VOICE service

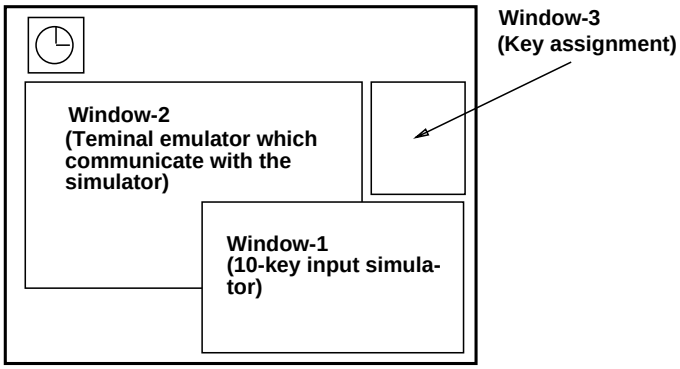

図 1.7: WIDE/Phone-shell シミュレータ

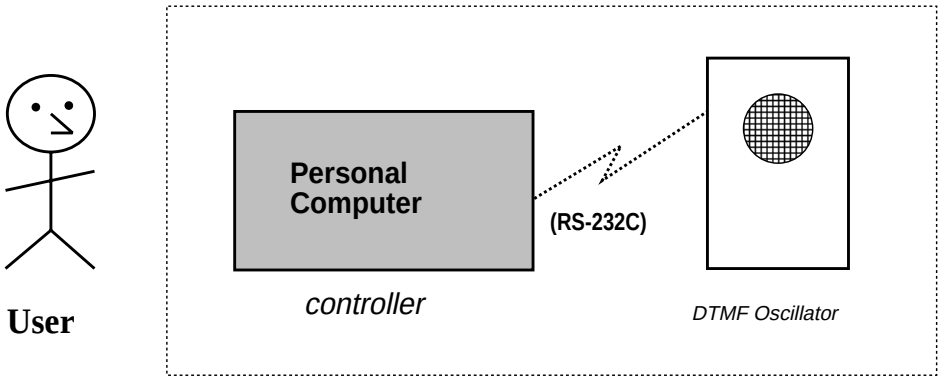

図 1.8: 試作版ダイアラの利用形態

```
% text2tex foo > foo.tex
% jlatex foo.tex
% lpr -fax -Pfax -T03-123-4567
```

図 1.9: ファクシミリへの出力

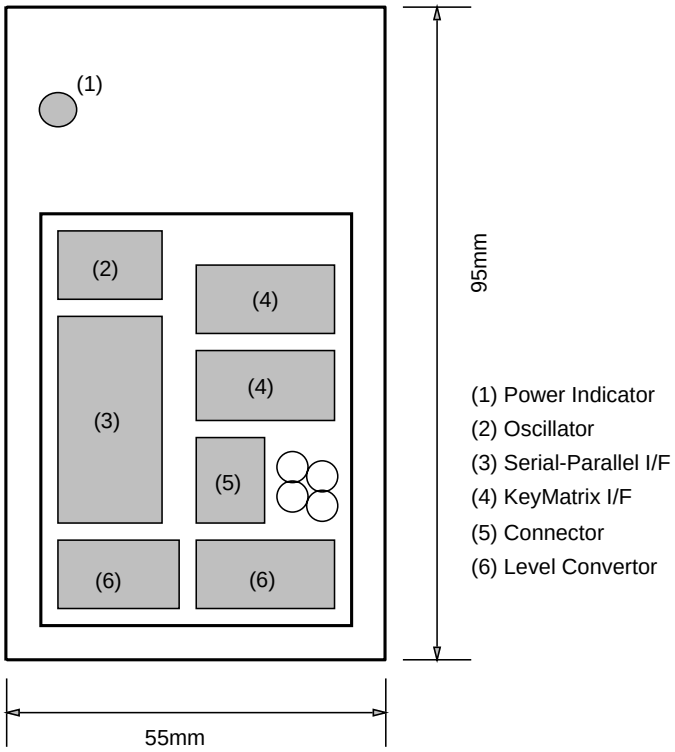

図 1.10: 試作版ダイアラの構成