インターフェイスの街角 (1)
携帯情報端末 Pilot のプログラム開発

増井俊之

今月から、「インターフェイスの街角」と題して、ユーザー・インターフェイスにまつわるさまざまな話題を紹介していきます。

今回は、携帯情報端末（いわゆる PDA）の Pilot をとりあげ、そのアプリケーション開発環境とユーザー・インターフェイスについて解説します。

Pilot

3Com（旧 U.S.Robotics）の「Pilot[1]」[1]という携帯情報端末が人気を集めています（写真 1）。日本でのシェアはザウルスなどにくらべてまだまだ小さいものの、ある雑誌のアンケートで「いま、もっとも推薦に値する携帯端末」としてパワーザウルスや Libretto を抑えて 1 位にランクされたこともあります。調査会社 DataQuest によれば、発売元の米国の市場では携帯計算機の 50%、PDA の 70%のシェアを占めているそうです。最近、IBM も米国で OEM 販売を始めました。現時点では日本語版は市販されていませんが、システム・ライブラリを拡張して日本語の入力や表示ができるようにした「J-OS[2]」を使えば日本語も扱えます（発売元はイケショップ[3]）。

Pilot の人気の秘密は小さい (12.0 × 8.1 × 1.8cm)、軽い (160g)、速い、安い（米国では 300 ドル以下）といった点にあるようです。とくにその小ささは特筆もので、ワイシャツの胸ポケットにすっぽり収まります。それにもかかわらず、PDA として実用的に使えるところが評判のよい理由でしょう。CPU には、MC68000 に PDA 用の周辺回路を付加した 16MHz の MC68328（通称 DragonBall）[4]が使われています。とくに強力ではありませんが、アプリケーションが小ぶりに実装されているため、小気味よく起動、反応します。

写真 1　PalmPilot

図 1　PalmPilot の Web ページ

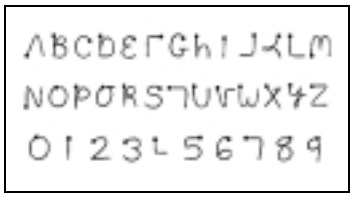
その他の仕様・特徴は以下のとおりです。

- 160 × 160 ドットのモノクロ液晶画面
- HotSync ボタンによる一発シンクロ
  パソコンなどのシリアルケーブルに接続した ``Cradle´´

1 1996 年 3 月に発売された Pilot1000（メモリ 128KB）と Pilot5000（512KB）、1997 年 3 月に発売された PalmPilot Personal（512KB）と PalmPilot Professional（1MB）の 4 機種があります。このうち、Pilot1000 と Pilot5000 はすでに生産を中止し、現在は Personal と Professional が販売されています。この 2 機種の違いは、メモリ容量と TCP/IP 接続の機能が組み込まれているかどうかの 2 点です。

写真 2　Cradle

図 2　Graffiti のペンストローク（一部）



という台に Pilot を置き（写真 2）、Cradle のボタンを押すだけで、Pilot 内部のデータとパソコン内のデータの同期をとることができます。

- Graffiti によるテキスト入力
  本体下部のタッチパネルで、英文字や記号に似た Graffiti という一筆書き記号（図 2）を描くことにより、アルファベットや記号が入力できます。
- 各種のアプリケーション
  電子メール、スケジュール、メモなど、携帯端末用のアプリケーションが揃っています。Web ブラウザなども開発されています。
- TCP/IP ライブラリ内蔵
  シリアル端子や外付けモデムを介して、簡単に外部との通信ができます。

Pilot のプログラム開発

このように、Pilot はユーザーにとって魅力的な特徴を備えているだけでなく、開発環境や資料が整っており、パソコンや UNIX マシンで手軽にアプリケーションが開発できます。そのため、趣味のプログラマーやユーザー・インターフェイスの研究者にとってもたいへん魅力的です[2]。PC 互換機や Macintosh 上では、Metrowerks の CodeWarrior という統合開発環境[5]が利用でき、一般の UNIX システムでも GCC などのフリーのツールを使って開発したアプリケーションを Pilot に転送して実行させることができます。

最近は、多種多様な携帯情報端末が開発され、市販されています。しかし、出来合いのソフトウェアにものたりなさを感じて自分でプログラムを作ろうと思っても、内部構造が公開されていなかったり、コンパイラなどのプログラミング環境が利用できなかったりと、プログラム開発は容易ではありませんでした。その点、Pilot は最初から内部ライブラリ仕様が公開されており、C コンパイラを含むプログラミング環境も市販されています。このオープンな方針のおかげで、数多くの趣味のプログラマーが Pilot のプログラムやゲームを開発し、Web ページなどで公開しています[6]。

Pilot のライブラリ

Pilot のアプリケーション開発に利用できるライブラリには、次のようなものがあります（これらに関する資料は、3Com の Web ページ上で公開されています）。

インターフェイス関連

Pilot の GUI アプリケーションは、デスクトップ計算機の場合と同じように、ペンや本体下部のボタンからの入力イベントに反応するインターフェイス・ツールを画面に並べて作ります。これらのツールとしては、ボタン、チェックボックス、テキスト編集枠、スクロールバーなどが用意されています。インターフェイス・ツールの位置や初期値などは、プログラムとは別にリソースとして定義します。このため、アプリケーション・プログラムから明示的にツールを呼び出したり、初期化したりする必要はありません。ライブラリの一部を以下に紹介します。

- イベント処理
  ペンタッチ入力や Graffiti での文字入力によるイベントやキューを扱います。
- ツールキット関連
  ウィンドウ管理、描画、メニューなどのインターフェイス・ツールの操作関数があります。

2 Pilot は、ペン入力や協調作業などの実験によく使われているようです。先日、カナダで開催されたユーザー・インターフェイスのカンファレンス (UIST 97) でも、参加者の多くが Pilot を携帯していました。

●Graffiti 関連
Graffiti の認識状態や認識辞書を管理します。
●サウンド関連

メモリ、データベース関連

Pilot には、UNIX や DOS のようなファイルシステムはなく、プログラムやデータ、スタック、動的メモリ領域はメモリ上のブロックとして表現されます。

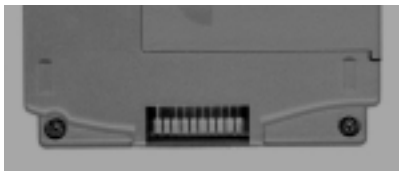
多くの携帯端末では、ディスクと同じようなファイルシステムをメモリ上に置いているので、アプリケーションの起動時に、ファイルシステムを構成するメモリから実行メモリにプログラムがコピーされたり、データを保存するときに、ファイルシステムのメモリにコピーする必要があったりといった無駄が生じてしまいます。

これに対し、Pilot ではどのようなプログラムやデータについても、メモリ上のものをそのまま使います。メモリ上のデータを扱うために、メモリ管理ライブラリとデータベース管理ライブラリが用意されています。

●メモリ管理ライブラリ
メモリブロックを作成/削除したり、サイズや属性を変えたりします。
●データベース管理ライブラリ
Pilot では、ファイルシステムの代わりにメモリ上の〝データベース〟が使われます。データベースは、メモリ管理ライブラリで扱えるメモリブロックの集合で、名前(文字列)またはタイプ(4 バイト)とクリエータ(4 バイト)との組合せによって参照できます。
データベースには、データ・データベースとリソース・データベースの 2 種類があります。前者のなかのデータはレコード番号によって参照でき、アプリケーション・データの保持などに使われます。後者は雑多なリソースデータをまとめて扱うためのデータベースで、各リソースはリソースタイプ(4 バイト)とリソース ID(4 バイト)によって参照されます。
Pilot アプリケーションは、リソースタイプが〝code〟のコードリソース(プログラム本体)や〝tFRM〟のツールキット・リソースなどを集めたリソース・データベースとして実装されており、専用のタイプ〝appl〟とアプリケーション独自のクリエータ ID[3]をもっています。個々のアプリケーションは、それぞれ異なるクリエータ ID をもつ必要があるので、開発者が Web ページで登録する仕組みになっています[7]。

3 たとえば、インベーダー・ゲームのクリエータ ID は〝INVD〟です。

写真 3　Pilot 本体の RS232C ポート



通信関連

Pilot のケースの下側には、むき出しの RS232C シリアルポートが付いています(写真 3)。プリント基板が唐突に突き出している感じなので最初は驚きますが、ほかの計算機とシリアル接続するための Cradle に置くだけで回路が接続され、通信ができるようになるので便利です。

シリアル端子経由で 1 文字ずつ入出力をおこなう普通の非同期通信ライブラリのほかに、PPP による TCP/IP ライブラリが用意されており、外付けのモデムを使って通信するプログラムなども比較的簡単に開発できます。

システム、数値演算、標準ライブラリなど

時計/アラーム関数、エラー処理、検索、パスワード管理、文字列操作、乱数、ソーティング、浮動小数点演算、時間計算などのシステム関数や標準ライブラリが用意されています。

◇◇◇

このように、システム・ライブラリや演算ライブラリからユーザー・インターフェイスまで、かなりの量のライブラリがきめ細かく用意されているので、標準的なアプリケーションの作成であれば、これらのライブラリだけでほとんど間に合うでしょう。

「Hack」プログラム

ライブラリ関数のエントリは、トラップテーブルとして並べられているので、その値を書き換えて、システムのライブラリ関数を自作の関数に置き換えてしまうことができます。使い方を誤ると大変ですが、システムの挙動を簡単に拡張できる便利な仕組みです。トラップテーブルを書き換える Pilot プログラムは、普通のアプリケーションと区別して〝Hack〟と呼ばれています。これらの Hack を有

効/無効にしたり、統一的に扱うための ``HackMaster'' というプログラムも開発されています[8]。HackMaster の取決めに従った Hack を使えば、ともすれば無法地帯になりがちなこの種のプログラムを、ある程度管理して使えるようになります。

J-OS は、Pilot の表示関数や文字入力関数などを日本語用に拡張した Hack として実装されています。たとえば、画面に文字を表示する関数 ScrDrawChars() を拡張し、引数が Shift-JIS コードの場合はアルファベットの代わりに漢字を表示します。

UNIX の Pilot 開発環境

さきほども書いたように、Pilot の内部ライブラリ仕様などに関する資料は数多く公開されています。さまざまなプラットホーム上で各種の開発システムが作られており、C、アセンブラ、BASIC、CASL、Scheme など、いろいろな言語を用いてアプリケーションを開発することができます。UNIX 上では、GCC を中心とする以下のツール群を使うことにより、完全な Pilot アプリケーションを手軽に開発できます。

GCC	コンパイラ/リンカ
obj-res	COFF からのリソース抽出
pilrc	インターフェイス用リソース・コンパイラ
txt2bitm	アイコンリソース・コンパイラ
build-prc	Pilot 用モジュール作成
pilot-link	データ転送

さらに、Pilot シミュレータである copilot や xcopilot などの補助的なツールを使えば、さらに効率的な開発も可能です。

これらのツールについて、もうすこし詳しく解説しましょう。

GCC

Pilot の CPU である DragonBall は、MC68000 のコアに携帯端末用の各種周辺回路を追加したものなので、コンパイラ自体は MC68000 用のものが使えます。Pilot 用の GCC は、gcc-2.7.2.2 にパッチを当てて作成します。このパッチと obj-res、pilrc、build-prc は、anonymous FTP サイトから入手できます[9]。

Pilot 用の GCC では次の点に注意が必要です。

- 整数は 2 バイト
  `sizeof(int)` の値は 2 です。
- 連続したメモリ領域は 64KB まで
  ポインタは 4 バイトですが、メモリブロックは 64KB 以下に制限されているため、64KB より大きな連続メモリ領域は使えません。
- 変数初期化の制限
  文字列定数の配列は初期化できません。つまり、

```
char *s = "abc";
```

は期待どおり初期化されますが、

```
char *s[] = {
        "abc",
        "def"
};
```

は動作しません[4]。

前述のように、Pilot ではプログラムやスタック領域などを含めてあらゆるデータがメモリブロックとして管理されています。しかし、メモリブロックが最大 64KB になっているため、それより大きな連続領域は使えません。たとえば 64KB を超える配列は作れませんし、スタックサイズもこれを超えることはできません。

これらの点に注意すれば、Pilot のアプリケーションも Macintosh や UNIX の GUI アプリケーションと同様な感覚で開発できます。

リソース抽出 (obj-res)

GCC でコンパイル、リンクをおこなうと、COFF 形式のオブジェクト・ファイルが生成されます。Pilot にプログラムやデータをロードするためには、これを Pilot のリソース形式に変換する必要があります。obj-res は、COFF 形式のオブジェクト・ファイルを Pilot のリソースに変換するためのプログラムです。

リソース・コンパイラ (pilrc)

テキスト編集枠やボタンなどのユーザー・インターフェイス部品は、プログラムやデータとは別のリソースとして定義します。CodeWarrior では、GUI を使って各種の

4 この問題は、コンパイラのバージョンアップによって解決する可能性があります。

リソースを定義、編集します。これに対し、UNIX では pilrc (Pilot Resource Compiler) というリソース・コンパイラを用いて、テキストによるリソース記述ファイルから、各インターフェイス・ツールに対応する複数のリソースファイルを作成することができます。

アイコンリソース生成 (txt2bitm)

txt2bitm は、テキストエディタで作成したビットマップ・ファイルをアイコンリソースに変換するツールです。

モジュール作成 (build-prc)

ここまでに紹介したツールで作成した複数の関連するリソースファイルをまとめて Pilot に転送し、Pilot のリソース・データベースにすればアプリケーションが完成します。

開発マシン上でリソースファイルを結合し、`PRC 形式´のファイルを作成します。これを、HotSync によってシリアルケーブル経由で Pilot 本体に転送すると、本体上にアプリケーションのリソース・データベースが作成され、実行できるようになります。残念ながら、PRC ファイルの構造は 3Com からは公開されていませんが、ユーザーによる解析結果が公開されています[10]。

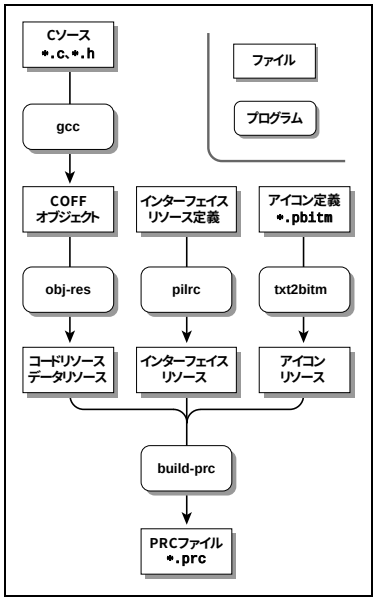
build-prc は、複数のリソースデータをまとめて PRC ファイルを作成するツールです。アプリケーションを転送するためには、タイプが `appl´の PRC ファイルを作りますが、ほかのタイプのリソース・データベースを作って転送することもできます。

データ転送 (pilot-link)

UNIX と Pilot とのデータ転送には、pilot-link というデータ転送パッケージを使います[11]。これはスケジュール・データなどの各種データの転送プログラムを含んでいますが、**pilot-xfer** というプログラムを使って任意の PRC ファイルやデータベース・ファイル（データ・データベースに対応し、`PDB ファイル´と呼ばれます）を UNIX と Pilot とのあいだで転送できます。つまり、プログラムの転送は、

1. build-prc で PRC ファイルを作る
2. pilot-xfer で Pilot 本体に転送する

図 3　GCC による Pilot プログラム開発の流れ



という手順でおこないます。pilot-xfer はプログラムを Pilot に転送するだけでなく、Pilot に入っている任意のデータベースを UNIX に転送することもできます。

pilot-link パッケージには、pilot-xfer 以外に、PRC ファイルからリソースファイルを抽出する（build-prc の逆の操作）**pilot-file** や、カレンダーファイルを抽出する **read-ical** などの有用なプログラムが含まれています。

プログラム開発の流れ

以上のツールを使った UNIX でのプログラム開発の流れは図 3 のようになります。

Copilot

高速なマシンで小さなアプリケーションを開発する場合、コンパイルから PRC ファイルの生成までは数秒で終りますが、ここまでに紹介した方法ではそのたびにプログラムを Pilot に転送して実行させる必要があります。Pilot のシミュレータ Copilot[12]を使えば、Windows や Macintosh 上でプログラムの動作を確認できます（写真 4）。Windows 用の Copilot を X11 に移植した xcopilot というシミュレータもあります[13]。

Copilot はたいへんよくできており、Pilot の実物を持っていなくても、Pilot のプログラム開発ができてしまう

写真 4　Copilot（Windows 上の Pilot シミュレータ）

図 4　Pilot プログラムの基本構造

```
PilotMain()
{
      if (アプリケーション起動){
            (アプリケーション初期化)
            while(EvtGetEvent()){
                  (イベントの種別に応じた処理)
            }
            (アプリケーション終了処理)
      }
      else{
            (検索などの処理)
      }
}
```

ほどです。

プログラムの構造

Pilot のアプリケーションは基本的に図 4 のような構造になります。

Application ボタンを押してアプリケーションの起動が選択されたり、Find パネルによりアプリケーション・データの検索が指令されると、アプリケーションの PilotMain() に制御が移ります。アプリケーションの起動が指示された場合は、必要な初期化をおこなったあとでイベント処理の無限ループを実行します。

簡単なプログラム例

リスト 1 は、ペンで画面をタッチすると、その場所に ``abc''という文字列を表示するプログラムです。

メインルーチン PilotMain() では、EvtGetEvent() でイベントを取得したあと、まずシステムイベント処理関数 SysHandleEvent() によって電源ボタンなどのハードウェア・ボタンや Graffiti 入力領域でペンをタッチしたイベントかどうかを判定し、その場合にはシステムが処理をおこないます。同様に、MenuHandleEvent() によってメニューイベントかどうかを判断したあと、ユーザー定義の ApplicationHandleEvent() で目的の処理を実行します。

おわりに

携帯情報端末 Pilot と、その UNIX 上でのプログラミング環境について駆け足で説明しました。掌の上で動くちょっとしたプログラムを作ってみたくなったのではないでしょうか。

UNIX 上の Pilot プログラミング環境はまだまだ発展途上で、頻繁にツールに新しい機能が追加されたり、新しいツールが発表されたりしています。開発システムの詳しい使い方や最新情報については、GNU 上の開発システムのチュートリアル[14]や Pilot-UNIX メーリングリストの記事[15]などを参照してください。

次回は、Pilot の開発環境を使って、もうすこし気のきいたインターフェイスを作ってみましょう。

（ますい・としゆき　ソニー CSL）

[参考 URL]

[1] PalmPilot
http://palmpilot.3com.com/palm/index.html
[2] YAMADA Tatsushi's Home Page
http://www.tt.rim.or.jp/~tatsushi/
[3] Ikeshop Pilot 団
http://www.ikeshop.co.jp/pilot/
[4] MC68328 DragonBall Microprocessor
http://www.mot.com/SPS/ADC/pps/prod/3XX/mc68328.html
[5] CodeWarrior for PalmPilot
http://www.metrowerks.com/products/cw/pilot/
[6] Pilot アプリケーションのアーカイブサイト
http://www.pilotzone.com/
http://www.pilotgear.com/
[7] クリエータ ID の登録ページ
http://palmpilot.3com.com/palm/devzone/crid/crid.html
[8] DaggerWare's HackMaster
http://www.daggerware.com/hackmstr.htm
[9] FTP area for the UNIX PalmOS/Pilot development project

リスト 1　ペンタップ位置に文字列を表示するプログラムと Makefile

●プログラム

```
#include <System/SysAll.h>
#include <UI/UIAll.h>

static Boolean ApplicationHandleEvent(EventPtr event)
{
        Boolean handled = false;
        if (event->eType == penDownEvent){
                WinDrawChars("abc",3,event->screenX,event->screenY);
                handled = true;
        }
        return handled;
}

DWord PilotMain(Word cmd, Ptr cmdPBP, Word launchFlags)
{
        EventType event;
        Word error;
        if (cmd == sysAppLaunchCmdNormalLaunch)
        {
                do {
                        EvtGetEvent(&event, evtWaitForever);
                        if (SysHandleEvent(&event)) continue;
                        if (MenuHandleEvent(NULL, &event, &error)) continue;
                        if (ApplicationHandleEvent(&event)) continue;
                        FrmDispatchEvent(&event);
                }
                while (event.eType != appStopEvent);
        }
        return 0;
}
```

● Makefile

```
CC=m68k-palmos-coff-gcc
CFLAGS = -O2 -g -Wall
OBJRES = m68k-palmos-coff-obj-res
BUILDPRC = build-prc

ICONTEXT = "Test"
APPID = TEST
test.prc: test
   $(OBJRES) test
   $(BUILDPRC) test.prc $(ICONTEXT) $(APPID) *.bin *.grc
```


ftp://ryeham.ee.ryerson.ca/pub/PalmOS/prc-tools.0.5.0.tar.gz

[10] ユーザーによる PRC ファイルの構造解析
http://web.mit.edu/tytso/www/pilot/prc-format.html
http://juju.cs.uec.ac.jp/pilot/prcmemo.html

[11] pilot-link
ftp://ryeham.ee.ryerson.ca/pub/PalmOS/

[12] Copilot
http://userzweb.lightspeed.net/~gregh/pilot/copilot/（Windows 用）
http://members.aol.com/illumesoft/copilot.html（Macintosh 用）

[13] xcopilot (Pilot stuff from the ISAAC Group)
http://www.isaac.cs.berkeley.edu/pilot/

[14] GNU Pilot SDK チュートリアル
http://www.iosphere.net/~howlett/pilot/GNU_Pilot.html

[15] Pilot-UNIX メーリングリスト・アーカイブ
http://www.acm.rpi.edu/~albert/pilot/