夏のプログラミング･シンポジウム「プログラム詠み会」2015.9.4-6

LaplasのRaspberry piへの移植

原田　康徳[1,a)]

概要：著者が学部生の頃に作成した記号処理言語 Laplas は，当時雑誌 ASCII に 4 回連載され，そのソースコードとオブジェクトがディスクアスキーとして販売された．言語は CONS セルをメモリ構造としてもつスタックマシンで，forth のようなシンタックスでリスト処理を行う．このソースコードはフロッピーディスクが読めないということでうちなわれつつあった．夏のプロシンの最中に，そのソースコードとたまたまネットで知り合った人から頂くことができたため，そのソースコードを Raspberry pi への移植を試みた．グラフィックの移植までは進まなかったが，簡単なプログラムを動作させるところまで行った．

キーワード：記号処理言語，スタックマシン，ラズベリーパイ

1. はじめに

著者は高専の卒業研究で 3 次元のタートルグラフィック（トンボグラフィック）を開発し（1983 年頃），プログラムで 3 次元の形を生成させていた．当時の流行はレイトレース画像を何時間もかけて計算するということであったが，著者はインタラクティブに生成することに面白みを感じており，画像はワイヤーフレームで即座に表示される計算を行っていた．トンボグラフィックは，Pascal で記述され，プログラムを修正しては表示させるということを繰り返していたことから，インタプリタ用の言語の開発が望まれていた．

北大に編入してから，山本強先生が Lisp の上でタートルグラフィックを動かしているデモを見せて頂いて．Lisp という言語の拡張性の高さと，インタプリタの楽しさに惹かれた．しかし，どうしても括弧だらけな構文は好きになれなかった．その以前に，著者は FORTH の移植を Z-80 上で（ハンドアセンブルで）行っており（未稼働）逆ポーランド記法の言語実装には明るかった．そこで，逆ポーランド記法のリスト処理言語をつくり，その上で 3 次元タートルを動かすという着想を得た．それが Lasplas[1] という言語の始まりである．Laplas は Language Processor for Listing And Stacking の略であり ce じゃなく s である．

Laplas は 2 つのバージョンが作られた．最初は Pascal の上で，コンスセルは Integer の配列として実装していた．実装は容易で，どんな環境にでも直に移植できた．しかし，当時 68000 や 8086 などへの Lisp の実装法が進み，8086 のセグメントを 2 つ使った 16 ビットのセルの実装や，68000 の上の 8 ビットが使われていないということを使ったポインタへのタグの埋め込みなど，興味の惹かれる技術が多かったため．それにならって C 言語での実装に切り替えた．後の ASCII に乗ることになったのもこの C 言語バージョンである．

また，当時学部 3 年であるにもかかわらず，コンピュータの沢山ある北村先生（高専編入担当）の研究室に出入りしていた．そこには様々な出入りの

[1] デジタルポケット
[a)] viscuit@gmail.com

企業（UNIVAC, ソード, toshiba, fujitsu, hitachi など）がつくったコンピュータがデモ機として置かれていた．まだ，PC9801 と IBMPC に二本化する前であったため，互換性がない機種であったが，ほとんどは MS-DOS や CPM/86 などの OS が動いていた．そのデモ機をお借りすることの引き換えに Laplas を移植して見せるということをしていた（させられていた）．

その後 unix 環境での動作も必要となり（appolo domain，UNOS など），ポインタにタグを埋め込む実装だけでなく，変なことをせずセル上にタグ情報を置く実装などいくつか用意していた．いずれにしても long は 32 ビットで short は 16 ビットで，まだ C++は登場していない．

Laplas には画面上でスクリーンエディタの機能をもっていた．80 文字 25 行の 1 画面分である．scsave というコマンドでそのコマンドより上の画面を保存，scload で保存された画面を読み出すことができた．インタプリタであるから読み込んだプログラムを実行する一連のコマンドとパラメータがあるが，その使用例を画面に読み出すことができ，そこのカーソルを移動して Return キーで実行させることができたのである．それらの移植のためには vt-100 互換のエスケープシーケンスを呼び出し，それで大体の機種は動作していた．

また Laplas の特徴である 3 次元のタートルグラフィックス「トンボグラフィックス」も，画面消去，線描画命令などを各機種ごとに移植していた．

このような事情から，移植のしやすいソースになっていた．

2. Laplas の言語仕様

Laplas は CONS セルをメモリーモデルにもつスタックマシンである．データ構造は，整数（24 ビット），実数，CONS，文字列，ATOM の 5 つである．2 つのスタックと 1 つのプログラムポインタをもっている．プログラムポインタというのはプログラムカウンタのことであるが，メモリーに配列は無いためポインタである．スタックはデータスタックとリターンスタックがあり，それぞれポインタが積まれている．プログラムはリストで表され CAR にあるデータを実行し，プログラムポインタは CDR に進む．プログラムポインタが NIL を指しているとき，リターンスタックから pop してプログラムポインタにセットする．

プログラムの実行規則は次の通りである．＊整数，実数，文字列，リストの場合は，それ自身をデータスタックに積む．＊ ATOM の場合は，それの定義を実行する．特別な ATOM は組み込みの機能を呼び出す．その他の ATOM はリストの定義をプログラムとして実行する．＊ QUOTE という特殊なリスト構造は，それが引用されている ATOM を実行せずにスタックに積む．

```
'double (2 *) .
```

これは Laplas の関数定義である．最初の'double は QUOTE の double であるから double という ATOM をそのままスタックに積む．次の (2 *) はリストであるからそのままスタックに積む．最後の . はピリオドという ATOM であり，この定義を実行する．ピリオドはスタックから 2 つ降ろして最初の ATOM に対して次のリストを定義として与える．このプログラムに対して

```
5 double ?
```

というのを実行すると，5 をスタックに積み，double を実行する．これは先の定義で 2 *となっているため，2 を実行し（2 をスタックに積む）次に*はスタックから 2 つの数値を降ろしてその積を計算しスタックに積む．ここでは 5 と 2 を降ろして 10 を積む．最後に，?はスタックに積まれているデータを降ろして，それを画面に表示する．

```
1 (2 *) 10 repeat ?
```

まず，1 をスタックに積み，次にリスト (2 *) をスタックに積み，10 をスタックに積む．repeat はスタックから数字とリストを取り出し，リストの中身をプログラムとして数字の回数だけ繰り返し実行する．ここでは 2 *という命令を 10 回実行するので

```
1 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * 2 * ?
```

と等価な実行となる．2 を 10 回掛けていることから 2 の 10 乗を計算している．

3. Raspberry pi への移植

ポインタのサイズなどの問題は多少あったが，それ以上に C 言語がかなり仕様が変わってしまっている点が面倒であった．それらを順次修正してゆく必要があった．面倒というよりは，昔はこんなに違っていたのかという点を面白がっていた．

グラフィックの移植はまだできていない．

ESC シーケンスの部分は考えないでも動作したようである．

4. プロシンハッカソンについて

幹事として準備から関わっていたが，当日の体調もすぐれないという理由もあり，あまり沢山のコードは書けなかった．一つの原因として考えられる点は，まったく新しい環境でのプログラミングは開発環境の構築に大きく手間取ってしまうためよくないということである．一方で周りに知っている人が沢山いるという状況ではその人たちに簡単に聞けるという利点もあるため，こういう時こそ自分が得意ではない開発環境に挑戦するのも一つの考えである．ただし，参加者のなかで会期中の時間を充実したコーディングで過ごせた人は，やはり慣れた開発環境を使用していたようである．

参考文献

[1] 原田康徳, 北村正直: **Laplas: マイクロ・コンピュータに適した新しい言語**, 北海道大學工學部研究報告 130:145-156, 1986.