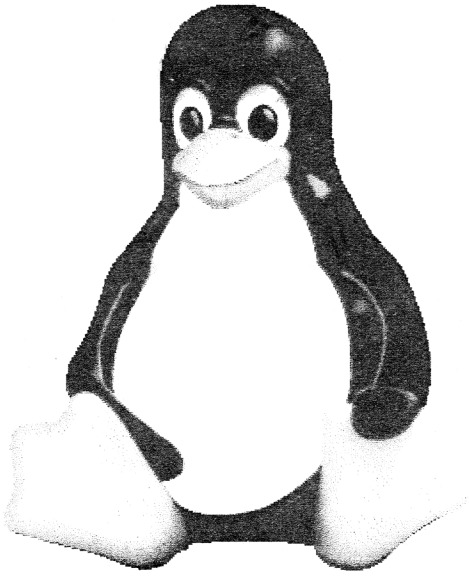
百万石

2004年 秋号

Linux 2.0 の公式マスコット
Tux くん。



Tuxくんは、Larry Ewing氏により
GIMPで描かれました。

MMA

百万石 2004年秋号 MMA

目 次

第 I 部　古典的な手口 **3**

1　Code Red **3**
1.1　アウトライン 3
1.2　具体説 3

2　SirCam **4**
2.1　愉快な伝播能力 4

3　(Distributed) Denial of Service Attack **6**

4　終わりに **6**

5　参考文献 **6**

第 II 部　IPnuts によるディスクレスルータの紹介 **8**

6　IPnuts とはどんなものか **8**
6.1　IPnuts とは 8
6.2　IPnuts の機能 8

7　実際に使ってみる **10**
7.1　ハードウェア 10
7.2　インストール 11
7.3　設定 11
7.4　おわりに 11

第 III 部　百葉箱プロジェクト **12**

8　インターネット百葉箱の視察 **12**
8.1　製品の構成 12
8.2　CC 版インターネット百葉箱におけるセンサ達 12

9　MMA 版インターネット百葉箱の構成 **12**
9.1　気候センサ 13
9.2　カメラ 13
9.3　データベースと Web インターフェース 14

10 現在の課題 **14**
10.1 センサ 14

第 IV 部　otenki Web インタフェース部 **15**

11 Web インタフェース部分 **15**
11.1 データの保存 15
11.2 Web インタフェース 15
11.3 おわりに 16

第 V 部　日本と海外の携帯電話の比較 **17**

12 価格 **17**

13 端末 **18**
13.1 Nokia 6630 18
13.2 Nokia n-gage 19
13.3 Nokia 6260 19
13.4 Nokia 7710 19
13.5 Nokia 9500 19
13.6 Nokia 7280 19
13.7 なぜ日本の端末と違うのか 19

14 将来展望 - 国の壁を崩せ！本当のグローバルスタンダードとは **19**

第 VI 部　デジタル温湿度センサの利用とマイコンによるその制御 **21**

15 ソフトウェアを作る前にやるべきこと **21**
15.1 LD スクリプト 21
15.2 スタートアップファイル 21

16 アプリケーションプログラムが行なっていること **21**
16.1 マイコンによるデジタルセンサの制御 21
16.2 PC へ測定値を転送する 21

17 付録:ソースコード **22**
17.1 LD スクリプト 22
17.2 スタートアップファイル 23
17.3 アプリケーションプログラム 24

第 VII 部　Pascal プログラミングの基礎 第一回 **31**

18 プログラムを書いてみる **31**

19 制御文 **32**
19.1 if 文 33
19.2 repeat 文 35

第 I 部
古典的な手口

(著者) 1 年[1]情報通信工学科[2][3] 橋本 達矢[4]

概要
始めまして。今年度新入部員の橋本です。昔流行った楽しげなことを持ち出して悦に入った経験をレジュメに綴らせて頂きました。懐古主義モードでないとネタが (略

1 Code Red

1.1 アウトライン

2001 年あたりに Code Red なるウイルスが流行っていたのだが、このウイルスの挙動を振り返ってみたい。また、Code Red には亜種が存在するがここではその内の Code RedII について取り扱った。
Code Red は Web サーバーである Internet Information Server[Services] のセキュリティホールを利用して不正な活動を行っていた。このセキュリティホールとは「Index Server ISAPI エクステンションの未チェックのバッファにより Web サーバーが攻撃される (MS01-033)」というものであり、起こりうる被害として、悪意あるコードを含んだ HTTP リクエストを送ることによりバッファオーバーフローを発生させてこの Code Red に感染するというウイルスであった。
Code Red は感染した計算機で下記のような挙動をとる。

- IIS の異常終了
- 広域スキャンによるネットワーク速度の低下
- コンテンツの改竄 (英語版でのみ発生した)
- トロイの木馬[5][6]の設置
- 他の IIS サーバーを探して伝染する

1.2 具体説

Code Red は Windows NT/2000 に感染し、特に 2000 の方での被害が尋常ではなかった。上記の挙動について詳しく見てみた。

1. バックドア[7]のインストール
   IIS のスクリプト用フォルダに cmd.exe[8]を C,D ドライブの特定のフォルダに root.exe という名前でコピーする。

[1]物理学年 (通算在籍学年)。本学の学生によってしばしば論理学年とのギャップがネタにされる。
[2]先輩曰く、学科再編時に他の学科の分野に分類できない分野の人材を集めて作られたいわば余り物を寄せ集めた中途半端な学科。
[3]近年、本学科名に限らず色物系の学部名や学科名に騙されて入学する学生が後を絶たない。
[4]MMA の同期から「薄い人間」と罵られている。
[5]古代の地中海地方にあったスパルタ国とトロヤ国が戦争した時にスパルタの名将ユリシーズがトロヤ攻略のために持ち出したでかい木馬。
[6]因みに神話の通り人間も入れるよう内部は空洞となっている。Sub Seven 等に代表される
[7]裏口。ある対象に対して最初のクラッキングの際次回以降の作業を簡略化するために設けられる
[8]サーバー上でリモートからコマンドを実行するために使われるプログラム。

2. トロイの木馬のインストール
   C,D ドライブのいちばん上の階層に窓使い[9]にはおなじみの explorer.exe を呼び出すプログラムである explorer.exe を作成する。この explorer.exe が次頁のような楽しげな挙動を取った。

3. レジストリの書き換え
   explorer.exe の呼び出しによりレジストリキー[10]

   ```
   HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\Winlogon
   ```

   のキー SFCDisable に値 0xFFFFFF9D をセットした。これによって窓の SFP[11]が無効化され重要なファイルの変更が可能となってしまった。

4. 仮想ディレクトリの作成
   explorer.exe の働きによって読み、書き、実行が可能となった仮想ディレクトリ C,D が作成される。これらの中身にはドライブ C,D がマッピングされていた。

5. 他のホストへの感染活動
   感染後、ほかのホストに対して感染活動を開始した。感染活動用スレッド[12]を 300 個作成し、それぞれのスレッドで 24 時間の間感染活動を行った。活動後にシステムを再起動し、最終的に Code Red がメモリから消えてバックドアの root.exe と explorer.exe が残った。この二つの実行ファイルが存在しているかどうか周りの人間が検索にかけて処置に困っていたのが記憶に残っている。

とまあ、ダラダラ蘊蓄をたれていますが、Code Red に感染して最も困ることといえば自分の意思と関係なく他のサーバーを攻撃してしまうことと、ホストアドレスがばれてしまった場合に他人から内部ファイルを操作されてしまうということが挙げられます。後者についての方法論は法律に触れるないしは種々の罪に問われる恐れがあるため提示しませんが[13]、当時の中流クラッカー達は楽しげに侵入していたようです。

2 SirCam

SirCam も Code Red と同じ様な時期に蔓延したウイルスでした。Code Red が NT 系窓 OS に感染したのに対して SirCam は窓 95/98/98SE/Me[14]に感染しました。感染媒体はメールの添付文書ファイルでありバイナリ切断を行うことで感染ファイルから文書部分のみを切り離すことも出来ました。

2.1 愉快な伝播能力

SirCam に感染すると下記のような挙動をとった。

1. `%TEMP%\<ファイル名>`と `C:\Recycled\<ファイル名>`として自分自身を複製。このファイルには添付された文書が含まれていた。

2. 自分自身を `C:\Recycled\Sirc32.exe` と`%System%\Scam32.exe` の名で複製

3. 値

   ```
   Driver32=%System%\scam32.exe
   ```
   をレジストリキー
   `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices`
   に追加

[9] BSD 信者から猛烈に蔑視される
[10] TEX の仕様で円印がバックスラッシュになっていますが、目を瞑って貰えれば幸いです。むしろ幸い推奨。
[11] システムプロテクション機能のこと
[12] スレッドと聞いて hoge ちゃんねるを思い浮かべたら変態扱い。マルチスレッド OS におけるプログラム実行の単位のこと。
[13] 要はチキン野郎ということ
[14] 窓の OS 史上最も重い物の一つ。取柄が無くてただ、ただ重くてどうしようもない。

4. HKEY_LOCAL_MACHINE\Software\SirCam

   というレジストリキーを作成し、下記の値をセット

   - FB1B - ごみ箱に保存された SirCam のファイル名を保存
   - FB1BA - SMTP IP アドレスを保存
   - FB1BB - 送信者のメールアドレスを保存
   - FC0 - SirCam の実行回数を保存
   - FC1 - SirCam のバージョン番号を保存
   - FD1 - 実行された SirCam の拡張子を除いたファイル名を保存
   - FD3 - SirCam の現在の状態に関する値を保存
   - FD7 - この処理の実行前に送信されていたメールの数を保存

5. HKEY_CLASSES_ROOT\exefile\shell\open\command
   というレジストリキーの値を
   C:\recycled\sirc32.exe "%1" %*"
   と設定した。これにより.exe ファイルを実行する度にワームが実行されるという楽しげな伏線が張られた。

6. ネットワークを通じて共有されている他のシステムを探し、発見すると、次の動作を行う

   - <コンピュータ名>\Recycled\Sirc32.exe に自分自身をコピーする。
   - <コンピュータ名>\Autoexec.bat ファイルに"@win \recycled\sirc32.exe"の行を追加する。
   - <コンピュータ名>\Windows\Rundll32.exe を <コンピュータ名>\Windows\Run32.exe にコピーする。
   - <コンピュータ名>\Windows\rundll32.exe を C:\Recycled\Sirc32.exe で置き換える。

7. 33 分の 1 の確率で自分自身を C:\Recycled\Sirc32.exe から %Windows%\Scmx32.exe に複写する。
   また、レジストリキー
   HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Startup
   が示す場所に自分自身を"Microsoft Internet Office.exe" として複写する。

8. 毎年 10 月 16 日になると、20 分の 1 の確率で C ドライブのすべてのファイルとフォルダを削除する。この発病症状は日付形式が D/M/Y（日/月/年）に設定されているコンピュータ上でのみ発生する。さらに、SirCam に含まれるファイルに FA2 という文字が連続して含まれ、そのすぐ後に sc という文字が続いていない場合、日付や日付形式に関係なく、すぐに発病処理が作動するという能力を持っていた。しかし、ランダム・ナンバー生成コードの初期化部分にバグがあったため、実際にはファイル削除およびスペース埋め尽くしの発病症状が発症する可能性はほとんど無かったと言って良い。

9. 上記の症状が発症した場合、C:\Recycled\Sircam.sys を作成し、ディスクスペースが
   なくなるまで文を追加する。文はは次の 2 つのうちのいずれか。

   [SirCam_2rp_Ein_NoC_Rma_CuiTzeO_MicH_MeX]

   または

   [SirCam Version 1.0 Copyright 2000 2rP Made in / Hecho en - Cuitzeo, Michoacan Mexico]

10. SirCam はメール処理に使う独自の SMTP エンジンを持っていて、次の 2 通りの方法でメールアドレスを取得する。

    - 次の 2 つのレジストリキーで参照されているフォルダから sho*.、get*.、 hot*.、*.htm ファイルを検索する。

      HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Cache
      HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Personal

      そして、見つかったファイルに含まれるメールアドレスを%Windows%\sc?1.dll ファイルにコピーする。
      ? の部分は、アドレスを取得した場所によって次のように異なる。

- scy1.dll: %cache%\sho*., hot*., get*. の場合
- sch1.dll: %personal%\sho*., hot*., get*. の場合
- sci1.dll: %cache%*.htm の場合
- sct1.dll: %personal%*.htm の場合

- %system%とすべてのサブフォルダにある*.wab (Windows Address Book すべて) を検索し、そこに登録されているメールアドレスを%system%\scw1.dll ファイルにコピーする。

11. 次のレジストリキーで参照されているフォルダから、拡張子が.doc、.xls、.zip のファイルを探し、

    HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Personal
    HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Desktop

    発見したファイルのファイル名を%system%\scd.dll に保存する。これらのうち、いずれか 1 ファイルが SirCam のオリジナルの実行可能プログラムに付加され、それがメールの添付ファイルとして使用されることになる。

    また、差出人欄のメールアドレスとメールサーバはレジストリキーから取ってくる。メールアカウントが存在しない場合、現在のユーザ名に"prodigy.net.mx"を付加する。例えば、JSmith としてログオンしている場合、SirCam によって使用されるメールアドレスは、jsmith@prodigy.net.mx となる。その後、SirCam はレジストリから取得したメールサーバか、次のいずれかのメールサーバに接続を試みる。

    prodigy.net.mx
    goeke.net
    enlace.net
    dobleclick.com.mx

3 (Distributed) Denial of Service Attack

某国のクラッカー集団がターゲットに対して今でも用いる手段。Distoributed と言う形容詞が付くことによって多対一での攻撃をする手法となる。高速回線が主流となった今日では常に大量のパケットを流し続けられるようになった為、ボトルネックの付近のネットワークがパケットで一層飽和しやすくなった。UNIX 系 OS を用いる物にとって DoS 攻撃はコマンドやスクリプトで容易に行えるため貧弱な実験台サーバーは簡単にフリーズして間抜けな反応を返してくれていたと技術者として働く恩師は語る。

4 終わりに

星の数ほどいる研究者や学者[15][16]が述べる所に因れば、現代の日本は素晴らしいピッチで情報化社会[17]になった産物であるが、ユーザーインターフェイスや利便性を革新するあまり情報自体の管理が行き渡っていない物だという。例えば警察とか。警察とか。警察とか。古いイデオロギーにとらわれた堅固な派閥体制とか官僚なんかに於けるヒエラルキーをもってしてもこの程度のもんです。公共機関側がステレオタイプに「お役人やお上は鈍くて頭が弱い」とレッテルを貼られる風潮は 100 年経っても変らないだろう。

5 参考文献

- http://www.ipa.go.jp/security/ciadr/vul/20010727codered.html
- http://www.symantec.com/region/jp/sarcj/data/w/w32.sircam.worm@mm.html

[15] 理論と結果が辻褄が合う物であるものなら、こじつけて自分の説として唱える人種。メンツを保つためなら学生といえども他人を平気でけなす。
[16] また、大学にいないと食べていけない職業であるため、高値で本を著したりして一般人に迷惑を掛けている事をしらない。
[17] 養老孟司は自己をいつになっても変わらないもの、即ち情報と成す社会と定義した

- 『バカの壁』著 養老孟司 新潮新書

第 II 部
IPnuts によるディスクレスルータの紹介

(著者) Hiroshi Fujita
pokka@mma.club.uec.ac.jp

概要
ルーターは最近、ブロードバンドの普及とともに大きく普及しているが、その機能は非常に限定されたものであり、パワーユーザには不満な点が多い。かといって PC を立ち上げると、さまざまな弊害がある。電源事情、ファンなどの騒音などがあるだろう。HDD を使うことでディスクトラブルもあるだろう。それを解決する CompactFlash でのルータについて書く。

6 IPnuts とはどんなものか

6.1 IPnuts とは

IPnuts とは、ネットワークアプライアンス用に作られた Linux の商用ディストリビューションである。1FDLinux などに触れたことのある方は、"Mosquito" というディストリビューションをご存知だろう。この Mosquito のニューバージョンが IPnuts である。

現在では 1FD ではなく、さまざまなパッケージの需要があり、2FD になっている。これはフリーでダウンロードできる。これでも個人用途では十分であるのだが、さらに高機能なものがパッケージされたものは有料となっており、これは FD ではなく CD からのブート、CompactFlash からのブートなどが可能である。

FDD にしても CD-ROM にしてもディスクであり駆動物である以上、トラブルになる可能性が高い。そこで今回は CompactFlash を用いてみる。CompactFlash は書き換えの回数に制限があり、普通に Linux を用いるとすぐに壊れてしまう。しかし IPnuts はブートにのみ CF を使い、ファイルシステムなどは RAM ディスクに構築するのでそういった心配が要らないというのも IPnuts の利点である。

6.2 IPnuts の機能

主な特徴、機能を紹介しよう。http://www.s-me.co.jp/products/ipnuts4_table.html より抜粋。

- ギガビットイーサネットの LAN カードを使用して、ギガビットルータを構築できます。
- 99 個までのインターフェースをサポートします。
- 簡単なソースルーティングや、ポートを指定したポリシールーティングができます。
- コネクショントラッキングにより SPI 機能をサポートします
- 複数インターフェース設定時に、インターフェースをグルーピングしてフィルタリングの設定ができます。
- PPPoE や VPN の仮想インターフェースなど、動的に作成されるインターフェースに対してルーティングが設定できます。
- スタティックルーティングのメトリック設定により、PPPoE 回線のアクティブスタンバイ構成ができます。
- ポリシールーティング設定時に、複数のゲートウェイを設定することで、ゲートウェイの負荷分散ができます。
- 送信先アドレス変換によって、IP アドレスまたはポートベースでサーバーへの負荷分散ができます。
- パソコン本体にキーボード、ディスプレが無くてもシリアルケーブル（クロス）で接続し操作できます。

- Web ブラウザから管理設定が行えます。
- 指定した IP アドレスに PING を実行し、返答が無い場合に管理者宛てにメールを送信します。
- syslog にログを保存します。リモートの syslog サーバーに転送することもできます。
- DNS サーバーとして機能し、クライアントからの要求を本体に設定されている DNS サーバーに問い直して返答します。
- スタティックルーティングと RIP をサポートしています。また、zebra(17) により、OSPF、BGP などの設定も可能です
- 高度な IP パケットのフィルタリング機能を持ちます
- 複数のパソコンを 1 台のパソコンに見せかけて通信することで内部のパソコンをインターネットから保護します。
- 外部からのアクセスをプライベートアドレスの公開サーバーに許可します。
- 公開サーバー専用のセグメントをプライベートアドレス、またはグローバルアドレスで作成できます。
- 擬似的なブリッジとして機能します。IP ネットワークを分けることなく、2 つのセグメントを接続します。
- ひとつのインターフェースに複数のアドレスがつけられます。NAT による異なるアドレスの公開サーバーの設定も可能です。
- DHCP サーバになります。クライアントの IP アドレスなどを自動的に割り振ります。
- DHCP クライアントになるので、CATV や ADSL の DHCP での接続ができます。
- PPPoE クライアント機能により、ADSL や光ファイバーなどの PPPoE での接続ができます
- 複数のインターフェースをひとつのブローキャストセグメントにし、かつフィルタリングを設定できます。
- tag-VALN 対応のスイッチングハブと組み合わせると、ひとつのインターフェースを複数のインターフェースとして使用できるようになります。
- 802.11b の無線 LAN のアクセスポイントにできます。（Prism の HOST AP モード対応カードのみ）
- DHCP リレーサーバになります。
- DHIS クライアントによりダイナミック DNS に対応します。
- これまでの Linux では不可能だった設定の初期化がコマンドを実行することなく初期化フロッピーによって簡単に行えます。コンソール不要。
- 複数の設定を保存でき、簡単に切りかえられます。(17)
- TTCP による対向側との通信速度の測定ができます。（コンソールから）
- フィルタリング設定による syslog と, コンソール上での tcpdump(17) によってパケットや通信の監視ができます。
- リモートからのコンソール操作を暗号化します。また scp によってファイルの転送もできます。(17)
- 本体の Web 管理画面を暗号化します。
- 自己認証で、x 509 証明書を発行する認証局になります。

- snmp のエージェント機能でルータを監視ができます。(17)
- IPsec で LAN 間接続 の暗号化 VPN を構築できます。AES 対応。x 509 対応。NAT Traversal 対応。回線冗長化をサポート。
- Windows から IPsec でポイントツーポイント接続し、リモート VPN 接続します。
- CIPE で LAN 間接続の暗号化 VPN を構築できます。UDP でトンネリングするためアドレス変換されている環境でも使用でき、かつ高速です。Blowfish で暗号化
- パソコンから直接本体と接続し、PPTP で暗号化 VPN を行います。クライアントライセンスは不要です。
- NAT 環境から PPTP の接続を可能にします。
- 認証、暗号化のない VPN ですが、マルチキャストを通します
- IPsec でカプセリングしで GRE の接続を行います。OSPF と組み合わせると、ダイナミックルーティングでの回線の冗長化が行えます。
- OSPF で ECMP をサポート。GRE over IPsec にて回線の負荷分散ができます。
- 生死確認をしながら、Web サーバーなどの負荷分散を行います。
- 生死確認時に、HTTP のステータスコードを指定できます
- VRRP で仮想ルータを設定し、IPnuts のハードウェアフェールオーバーを実現します。
- VRRP の状態変化によって、コマンドを指定し実行できます。
- PPPoE 接続の場合でも PPPoE の接続、切断を指示し、ハードウェアフェールオーバーします。
- IPsec 接続時にハードウェアフェールオーバーを実現します。

7 実際に使ってみる

7.1 ハードウェア

ハードウェアは普通の PC でかまわない。特に高速なものでなくても、CPU が 500MHz、メモリは 64MB 程度あれば実効スループット 80MBps 程度をたたき出してくれる。メモリが多いほうがコネクション数があげられるので 128MB くらいあればよいだろう。

今回は、ルータ用途ということで常時電源を入れるものなので低電力、低発熱なものとして、VIA C3 を使った。CPU は VIA C3 1GHz、メモリは 512MB を積んだ。

IDE に CF をつけるにはアダプタが必要だ。IDE-CF アダプタ などという名称で売られている。USB ブートができるものならば、USB のカードリーダでもかまわない。

ComapctFlash は SanDisk の 128MB のものを選んだ。最近の 10 倍速などとうたっている CF は IDE デバイスとして動かないことがあるので注意が必要。128MB でなくてもかまわない。64MB あれば十分な運用ができる。なお、CompactFlash は FAT16 でフォーマットすることが必要。Windows2000 や XP でフォーマットすると FAT32 になってしまい認識できなくなる。

[17] Mosquito を除きます。
[17] 機能の有無については一覧を参照してください。
[17] zebra は無償サポート対象外です。

7.2 インストール

実際、IPnuts の CF 版は CF パーツつきで販売されているが、これは CD-ROM 版からも製作可能である。CD-ROM 版の CD-ROM イメージのうち、/package の中身と/boot の中身のうち、288,img を除くものをコピーする。ディレクトリごとコピーしてはいけない。中身のみをコピーする。これだけである。

次に、syslinux.cfg の編集をする。cdboot という項目を消去なりコメントアウトし、boot=/dev/sda1:vfat を CF のデバイス名に変える。今回は boot=/dev/hdc1:vfat にする。

これができたら、CF を本体にさして、まずは CD で Boot する。いきなり CF での Boot はできない。CD で Boot したら、

```
syslinux -s /dev/hdc1
```

を実行する。これで CF からの Boot が可能になる。

CD を抜いて再起動し、CF から Boot すれば成功である。

7.3 設定

基本的に設定は Webmin で、Web ブラウザから設定可能である。もちろんコンソールでも設定可能である。だが、詳しくルーティングなどがわかっていない人にとって、コマンドでルールを書いていくのは難しいだろう。Web ブラウザからの動作設定は、市販のルータでもよく見かけるので設定は容易にできると思う。

詳しい設定方法は、IPnuts の説明書を見てもらえば十分であるので個々では割愛する。

7.4 おわりに

Linux や BSD を使う人にとって、フリーであるという理由は大きなものであるかもしれない。その中であえて有料の Linux を紹介したのは、十分に価値があるからだ。

機能を限定した、FD の IPnuts はフリーで手に入るので、使ってみてよいようならフルパッケージを利用してみたらどうだろうか。

第 III 部
百葉箱プロジェクト

(著者) oku

概要
百葉箱プロジェクトとは、現在総合情報処理センタ (以下 CC) で運用されている [1] インターネット百葉箱に相当するものを MMA で作ってしまおうというプロジェクトである。

8 インターネット百葉箱の視察

まずは敵を知るべしということで、今年の 5 月に CC のインターネット百葉箱を見学させて頂いた。

8.1 製品の構成

CC のシステムは内田洋行のインターネット百葉箱という製品で、teiten-cube と呼ばれる[18]Linux-PC と DAVIS 社の GroWeather 気象センサ [2] を中核に構成されている。CC のある総合研究棟の 10 階 (最上階) に teiten-cube は設置されており、これによってお天気情報の Web ページを生成し、発信している。

総合研究棟の屋上には、GroWeather が配置されており、teiten-cube とネットワークで結ばれている。これらのセンサとサーバはセットで提供されるもので、独自に組み合せるような構成要素は無い。

基本的に、インターネットに向けて公開されているホストは一つで、電通大関連の Web ページは CC に設置されている delegate によって中継されている。

8.2 CC 版インターネット百葉箱におけるセンサ達

GroWeather は、風速計、太陽放射強度と温度と湿度と紫外線強度センサ、雨量計をまとめた筒、カメラと制御装置を格納した箱で構成されていた。

温度計などは、地面からの放射の影響を抑えるために、下に人工芝が敷かれていた。風速計は少し離れたところに設置されていた。ビル風の影響もありあまり正確な観測は行えていないようだ。

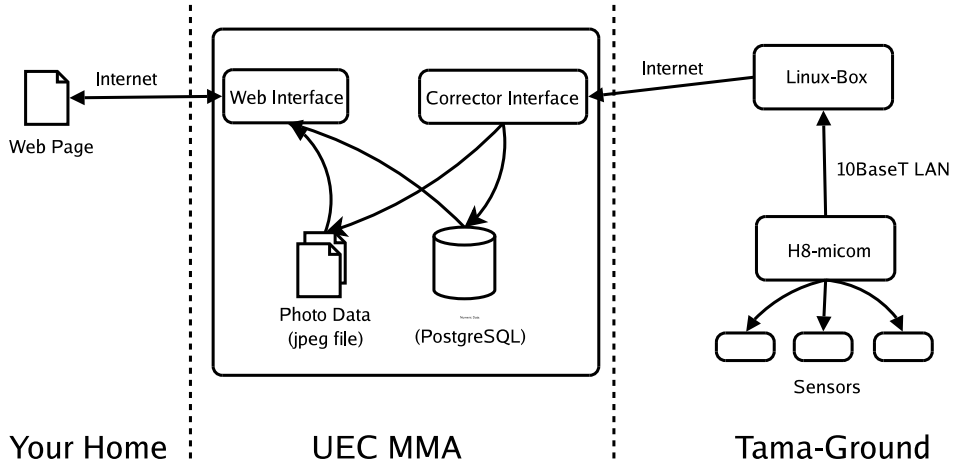
カメラはデジタルカメラに独自のファームウェアを乗せているらしく、デジタルカメラの画像を直接とりこめるようになっているらしい。

ちなみに、GroWeather は既にディスコンとなっている。

9 MMA 版インターネット百葉箱の構成

構成としては下の図のようになる予定となっている。おおよそ、CC のインターネット百葉箱と同じような構成となっている。

[18]正式名称なのかは不明



構成時に考えたのは、

- センサを実装した部分から実験出来るように
- なるべく汎用部品を使い、安く。
- 観測地に設置するシステムはなるべくメンテナンスせずに済むように。

設置場所として提案されていたのが、多摩川グラウンドと菅平宇宙観測所[19]で、さすがに何かあったときにメンテナンスに行かなければならないというのは困るので、観測地における機器は少くともディスクレスであることが求められた。

実際には、秋月の H8 マイコンキットのなかに LAN を装備したモデルが存在するので、それを利用することで Linux-box は廃止できそうだが、要求事項としてカメラ画像を得ることが有ったので、Linux-box の存在が前提となった。

9.1 気候センサ

CC の百葉箱はちゃんとした気象観測のためのセンサを備えているが、そういうセンサは大概かなり高価で、入手性もあまり良いとは言えないので、いわゆる機器制御用のセンサ IC をマイコンで読みとって利用することとした[20]。

まず検討したのは秋月で売っているような安価なアナログ読み取りのセンサで、これらは非常に安価ではあるが精度や対応範囲に問題があるので却下された。

今のところ実装をすすめているのは、Sensirion 製の SHT71 温度湿度センサ [3] で、これはシリアル I/O を利用して観測値を直接デジタルで得ることができ、それなりに対応範囲などの要求を満たしている。

9.2 カメラ

カメラはまだ決定事項ではないが、3 方式を検討している。

一つめは、USB カメラを利用する方法で、これは、カメラによってプロトコルが異なるのと、得てして USB のカメラは画質がよろしく無いというところにある。CC のようにデジタルカメラからの画像を入力することが出来るのなら、この方法は依然有力な候補足りうる。また、いわゆる Web カメラではかなり動作実績やコードの蓄積があるので、実現も比較的容易であると考えられる。制御用のライブラリは利用されているチップに合せていくつかの種類が存在する。

[19] 多摩川グラウンドは徒歩 30 分ほど、宇宙観測所はとりあえずかなり遠い

[20] もちろん、PC で直接読み取ることのできるようなセンサも存在するが、そもそも PC ではインターフェースの絶対的な数が不足している

もう一つは、IEEE1394 に対応したカメラを利用する方法で、1394 にはスタンダードのプロトコルとして、IIDC というプロトコルが有る。IIDC は、リアルタイムな非圧縮動画を転送するためのプロトコルなので、iSight のような Web カメラから比較的高級なカメラまで対応しているので画質や実現性の面ではかなり有力な候補である。また、IIDC はイメージの取り込みだけでなく、フォーカスのようなカメラの制御まで包括したプロトコルであることも利点である。ただし、「小さな」Linux システムには殆ど 1394 は搭載しておらず、外付けの PCI カードが必要になると考えられる。制御用のライブラリとしては libcam1394[4] が有る。

最後は、NTSC カメラを利用して、bt8x8[21]のようなビデオキャプチャデバイスから画像を取り込む方法で、これは比較的容易に実現できるが、現地の Linux-PC に通常の PCI バスを要求するので、システムが大きくなりがち。また、画質も NTSC カメラ程度の物しか望めず、bt8x8 キャプチャカードが長期間に渡って安定動作するのかどうかに疑問が残る。

9.3 データベースと Web インターフェース

観測結果を表示する方法としては、Web によるインターフェースを提供することとなる。また、データを蓄積する必要があり、データの平均やグラフを作成する必要から、数値データは RDBMS に格納する。

まず、観測装置のインターフェースと、Web インターフェースで表示するデータを合せるために、あらかじめデータベースのテーブルを定義してから両者を実装するという流れとなっている。データベースの定義を明確に行っているため、この両者は担当者が異なっているにもかかわらず、片方の変更は他に影響しない[22]。

10 現在の課題

10.1 センサ

とりあえず、他のセンサについても情報を収集する必要があるが、いくつかの自作困難なセンサが有る。

筆頭にあげられるのが雨量計で、人の目でたまった雨量を目視で確認して報告するならともかく、自動観測においては比較的複雑な機構をもった機械的なセンサをつくる必要がある。

実際の自動雨量計では、ちいさな升を機器の中に用意し、升の中に一定量の水がたまったら升が傾いて排水し、排水回数をカウントする仕組みになっている。

また、雪が降った場合の配慮のために融雪機を設置し、融雪機によって観測に影響が出ないようにする必要がある。

機械的な気象観測装置といえば、風向計も簡単に作ることは出来ない。どちらも気象観測の定番であるだけに、購入する必要があるだろう。

また、センサの設置についても防水など技術的な課題は多く残っている。

参考文献

[1] http://weather.cc.uec.ac.jp

[2] http://www.davisnet.com/weather/products/groweather.asp

[3] http://www.syscom-inc.co.jp/syscom_home_063.htm

[4] http://limu.is.kyushu-u.ac.jp/~yosimoto/ieee1394/

[21] いわゆるソフトウェアエンコードな安いキャプチャカードの類

[22] はずである

第IV部
otenki Webインタフェース部

(著者) Hiroshi Fujita
pokka@mma.club.uec.ac.jp

概要
Webインターフェースは PostgreSQL に入力されたデータをさまざまな形、日ごと、月ごとといった形で PHP で取り込み、グラフとして表示させせます。

11 Webインタフェース部分

11.1 データの保存

データ保存形式として、ファイル形式、MySQL、PostgreSQL とありましたが、ファイル形式では大量のファイルを端から端までなめるといった形になり、速度面で不利であるので SQL を使うことになりました。PHP は SQL と親和性があり、ライブラリがあるので簡単にアクセスができます。

MySQL と PostgreSQL のどちらにするかを考えたとき、MySQL は高速で、PostgreSQL は安定な SQL であるのでどちらを取るかを非常に悩みましたが、安定面を重視し、PostgreSQL にしました。

データベーステーブルは以下のように設定されています。

格納するデータ	レコード名	型
日付	date	timestamp
気温	kion	float8
湿度	situdo	float8
降水量	kosui	float8
風向	fuko	float8
風速	fusoku	float8
気圧	kiatu	float8
紫外線	sigai	float8
太陽放射強度	taiyo	float8

11.2 Webインタフェース

本来は、weather.cc.uec.ac.jp を参考に作るものでしたがプルダウンメニューやラジオボックスなどといったものを多用し、目的の出力を表示するまでに多くの手数がかかりどうも使いにくいというイメージを私が持ったので直感的操作で操作ができることを前提に、作り出しました。

基本的には 2-Click で目的のものが表示できるようになっています。”日付”と”対象”をクリックするだけでグラフが出てきます。

また、月別表示では SQL を使う強みを大きく発揮します。1 日のデータを平均し、それを日付の分だけ実行するわけですからファイル保存にしていたら、30 回ほどファイルを読むことになります。これに比べ、SQL ならば AVG() 関数を使うことで平均を瞬時に表示します。これが SQL を使った大きな理由です。

また、グラフ表示には jpgraph を使っていますので、グラフ大きさや目盛りの振り方、背景などのグラフのほとんどの項目を自由にカスタマイズできます。

11.3　おわりに

Web インタフェースについては、記事にするより使ってもらったほうがよくわかると思います。まだ未熟な点も多いですが、感覚的な操作ができることはわかってもらえると思います。

日付表示部のカレンダーは php.s3.to のものを使わせてもらっています。

第 V 部
日本と海外の携帯電話の比較

(著者) 情報通信工学科 niryuu

概要

日本は世界でも有数の携帯電話が普及している国だ。高校生のほとんどがケータイを持ち、写メールで画像を送信し、電車内でアプリで暇をつぶしている。日本では携帯電話は単なる電話を超え、必須の情報ツールとしての地位を確立した。これからも新サービスや機能が次々と誕生し、携帯は発展していくはずだ。

さて、驚くことに日本の携帯は世界的に見ると極めて異質な存在である。世界の常識が日本ではまったく通用しない。今回は日本の携帯が海外とどう違うかに付いて価格、端末、将来の展望という 3 つの側面でまとめてみた。

12　価格

海外の携帯は安い。そして、高い。日本の携帯の価格と比べると狂っていると言えるほどの価格差が見られる。通話料で 1 分 10 円レベルもあるのに端末は最低で 1 万、高いものだと 10 万を越える。市場原理で言えば先進国では余り格差がないはずの通信料金だが、どうしてこんなに差があるのか。その答えはビジネスモデルの違いにある。

日本では、サービスを行うドコモなどの通信キャリアが端末まで販売し、ヨドバシカメラなどの販売店に卸している。端末は電気店などで売っているが、家電などとは全く違った、キャリアとの契約込みでしか購入できない形式になっている。

キャリアは、契約台数を増やすために販売店に、「契約が取れたら報奨金を出す」ことを始めた。かなりの金額を出すので販売店は端末の代金を下げても、極端に言えばただで配っても利益が出るようになった。キャリアは客に長く料金を払ってもらいたいので、一年契約などが取れたらさらに報奨金を出した。この関係で早く解約したら違約金を払うように強要する販売店もあるが、法的拘束力はないので無視できる。

こうして日本の携帯の価格は安くなった。。最近は端末の高機能化から価格が上がり 2 万円台の物も出てきたが、それでも安い。さて、キャリアは販売店に莫大な報奨金を出しているが、その金の出所はどこだろうか。もちろんユーザーからの通話料からである。だから日本の携帯の通話料は高いのだ。このように契約する時は安く抑えて通話料で元を取る方式を「インセンティブモデル」という。海外でも多くのキャリアが採用しているが、ここまで徹底されているのは日本だけだ。

日本でインセンティブモデルがここまで成功した理由は、通信方式の違いにある。携帯がデジタルになった当時、今の 3G のように多数の規格が登場した。ヨーロッパでは GSM 形式、日本では PDC 形式が誕生した。当時は特に規格を統一する必要もなく、敢えていうなら国策で推進していく程度なので、実質市場原理に任せる形になった。結果として PDC を採用した国は日本しかなく、ヨーロッパやアジア等ほとんどの国は GSM を採用した。ちなみにアメリカの場合は特殊で、アナログ携帯がしぶとく生き残ってしまったので GSM と後発の CDMA、それと AMPS（アナログ）がしのぎを削る状況となった。　さて、ほとんどの国が採用した GSM では最初から海外で使うことを想定し、海外ではその国のキャリアを使うなどの需要も考え、SIM カードという契約したキャリアと電話番号が入った IC カードが端末に組み込まれた。現在の 3G の FOMA カードなどはこれの拡張形だ。

そんなこんなで GSM の端末は原則どこのキャリアでも使えるので、端末を売ることと契約をとることはほぼ完全に分離された。キャリアにとっては端末を売っても簡単に他のキャリアに乗り換えられてしまうので別にうま味はなく、端末は家電や PC と同じ扱いで販売された。

一方日本の場合は SIM カードのような方式は一部自動車電話を除きとられず、端末は一つのキャリアでしか使えない。キャリアを乗り換えるのは比較的困難だ。したがってキャリアは契約をとるために端末を安く売っても特に損はない。結局インセンティブモデルは日本の通信事情に適した方式であった。ちなみに海外でも端末に SIM ロックという制限をかけると一つにキャリアでしか使えなくなるので、これを使ってインセンティブモデルを実現しているキャリアも結構多い。だが解約するときに解除できるのがほとんどで、日本ほどの悪質さはない。

さてこのインセンティブモデル、3G が普及していく今日どうなっていくのだろうか。既にキャリアは USIM カードを採用しているが、当然のごとく端末にロックを掛けているのでドコモの 901i をボーダフォンで使えるということはない。この体制はしばらくは続くだろう。

むしろインセンティブは構造上の問題で消滅する可能性が高い。そもそもインセンティブは契約をとるためのモデルである。契約数がある程度まで増えたら契約をとるのは難しくなり、インセンティブの利点が減少する。機種変更に対するインセンティブも限界がある。普通のユーザーはある程度の機能をもった端末が出ればそれで満足し、最新機能を追うようなことはしない。現に海外ではいまだにメール機能もない端末を使っている人が多い。どのくらいの人間に動画や Flash が必要なのか。キャリアにインセンティブを続ける理由は以前より必ず低くなっているはずだ。

13 端末

海外では、ノキアやモトローラ、エリクソンなどが多くの魅力的な端末を発売している。まずはいろいろな機種を見てみよう。

表 1: 端末

Nokia 6630

Nokia n-gage

Nokia 6260

Nokia 7710

Nokia 9500

Nokia 7280

13.1 Nokia 6630

待望の W-CDMA 対応スマートフォン。PDA として PC のようにソフトをインストールして使える。音楽からメッセンジャー、FPS、エミュレータまで思いのままだ

13.2 Nokia n-gage

ゲーム機として発売されたが携帯としての機能も抜群。いつでもどこでもネット対戦可能。国内でも手に入りやすい

13.3 Nokia 6260

日本でも人気のリボルバー携帯が海外進出。ノキアらしい仕上がりとなっている

13.4 Nokia 7710

超高機能携帯の決定版。でかい液晶と最高の機能を搭載。動画も高画質で見られる

13.5 Nokia 9500

できるビジネスマン必携の端末。端末をぱかっと開けるとキーボードが出てくる。ワードやエクセルの文書も編集可能

13.6 Nokia 7280

ボタンがほとんどない小型の筐体に奇抜なデザイン。とにかく目立ちたいならこれ

13.7 なぜ日本の端末と違うのか

結構面白い端末が出てきたが、実はほとんどが日本の端末より低機能なのだ。その理由はいくつかあるがまず海外では前述のようにインセンティブがないので、端末の価格が高くなってしまう。当然 10 万円もする高級機はあまり売れないので、機能を抑えた安めの端末が多くなる。

もう一つの理由は、日本ではキャリアが端末に対して強い権限をもっているからである。キャリアは新しい端末に、例えば全機種メガピクセルカメラなどの基準を儲ける。大体 901i などシリーズごとにどの端末メーカーも同じ時期に出すので、このような芸当が出来るのだ。キャリアは他のキャリアとの差別化を図るためにメガピクセルカメラやテレビなどを売りにする一方、新サービスを使ってもらって通信料やコンテンツの収入を多くするために新サービスはすべての端末で使えるようにする。この結果日本の携帯はほぼ間違いなく世界でもっとも高機能な端末といえるようになった。

海外には全体的にデザインが日本より秀逸であるものが多い。INFOBAR などで驚いているようなら海外携帯にもきっと興味をもつはずだ。日本で使える端末も増えてきたので今後の新機種を要チェックだ。

14 将来展望 - 国の壁を崩せ！本当のグローバルスタンダードとは

いままで「国内」と「海外」を分けて議論をしてきたわけだが、数年もしたらこのような分類は無くなっているかもしれない。

その起爆材となったのは第三世代携帯、いわゆる 3G だ。3G では以前とは違い IMT-2000 という世界標準規格を策定した。この規格は通信方式や端末の規格などを含み、これに沿っている以上はすべての国のキャリアで同じ携帯が使える。例えばテレビ電話などの形式も規格に含まれ、テレビ電話対応の 3G 携帯ならドコモでもボーダフォンでもハチソン 3 でもどの国のどの携帯にかけてもテレビ電話が出来る。

これにより日本の端末を海外で使うことも出来るし、逆に海外の端末を日本で使うことも出来る。W-CDMA の場合は GSM との親和性が高く、キャリアにもよるが国外の GSM を採用しているキャリアの SIM カードを 3G

対応の端末に入れ、日本に持っていくと使えてしまうなんてことも出来る。3G が普及してしまえば今までのような「携帯鎖国」状態から晴れて解放されるわけだ。

しかし、まだまだ障害もある。その一つがドコモの W-CDMA の互換性の問題だ。ドコモはいうまでもなく世界に先駆けて W-CDMA のサービスを開始したキャリアであるが、急ぎ過ぎて世界標準でない W-CDMA を開始してしまった。正式な規格として採用された「リリース 99」と言う規格がまだ固まっていない段階で網を整備してしまったわけだ。そのせいで FOMA は都内など一部を除くほとんどの場所で互換性を欠いてしまった。この件については近年中には解決が図られるだろう。

技術的な障壁以外の大きな問題として、世界的に 3G に移行しないのではないかという懸念がある。現在動画などの高速度を必要とするコミュニケーションが一般化している国はほぼ日本だけで、海外では未だにショートメッセージが主流である。このような状況では現在の GSM と比較したアドバンテージがあまりなく、また GSM の拡張である EGSM を利用した場合の通信速度は 200kbps と 3G に近づいている（実測値では完全に 3G の勝ち）ため、3G は海外で現在も苦戦を強いられている。今後はメガクラスの速度が出る HSDPA や EV-DO などの普及局面に入れば完全に GSM は時代遅れの技術になるため、それなりの移行がなされると思われる。

第 VI 部
デジタル温湿度センサの利用とマイコンによるその制御

(著者) J/0321012, 仲田将之

15　ソフトウェアを作る前にやるべきこと

アプリケーションソフトウェアを作成するには, その前に行っておくべき事が幾つもあります. 以下にそのうちの 2 点を挙げておきます.

15.1　LD スクリプト

ソフトウェアを作成するにはそのハードウェアのメモリ構成があらかじめ分かっていなければいけません. そのメモリ構造を記しておくファイルが LD スクリプト, またはリンカスクリプトと呼ばれるものです. このファイルにはベクタと呼ばれるメモリ領域が定義され, 割込みが起こればそこに記されたアドレスにあるプログラムが実行されます. マイコンの起動も割り込みの一つでベクタの先頭に記されたアドレスからプログラムは実行されます.

今回は GNU Compiler Collection, 略称 GCC を利用してプログラムを作成したのでファイルの記述法はそれに従います.

15.2　スタートアップファイル

マイコンが起動すればベクタに記されたアドレスからプログラムが実行されるのですが, アプリケーションプログラムを実行させる前に行なっておくことがあります. H8 においてプログラムの全ての情報は ROM に記録されていて, どんなデータも読み出し専用であり書き込みできません. したがって, データを書換えるには全て RAM にコピーしておく必要があります. また, BSS と呼ばれるメモリ領域に記録された全てのデータは 0 で初期化しておかなければいけません. これらの作業をしてアプリケーションプログラムを起動するプログラムがスタートアップファイルです. C 言語において全てのプログラムが main 関数から始まるのはスタートアップファイルがこれを起動するためです.

このプログラムは IWATA 氏 [2] が公開しているものを利用しました.

16　アプリケーションプログラムが行なっていること

16.1　マイコンによるデジタルセンサの制御

マイコンは I/O ポート 8 のピン 5,6,7 をそれぞれデータ出力, データ入力, クロック出力としてデジタルセンサと接続して, クロックに同期して 1bit ずつのシリアルデータをセンサとの間で送受信します. これを行なうのがプログラム内の関数 getByte, putByte です.

デジタルセンサに温度や湿度の測定値を出力させるという命令を送れば, デジタルセンサからクロックに同期した 2byte のデータがそれぞれ出力されます.

16.2　PC へ測定値を転送する

デジタルセンサから送られたデータは, 関数 transto dec によって 10 進文字列へ変換されて, 関数 send string によってシリアルポートから 1 秒間隔で出力されます.

左側が温度値 SOt で右側が湿度値 $SOrh$ です:

```
6064 1880
6064 1880
6064 1880
6066 1879
6065 1879
6065 1878
...
```

これらの値から摂氏 T, 相対湿度 $RHtrue$ へ変換するには:

$$T = SOt \times 10^{-2} - 40 \tag{1}$$

$$RHtrue = (T - 25)(8SOrh \times 10^{-5} + 10^{-2}) + RHlin, \tag{2}$$

ただし

$$RHlin = -2.8SOrh^2 \times 10^{-6} + 4.05SOrh \times 10^{-2} - 4. \tag{3}$$

したがって, 出力値が $(6064, 1880)$ であるとき, 摂氏 $T0$, 相対湿度 $RHtrue0$ は

$$T0 = 60.64 - 40.00 = 20.64, \tag{4}$$

$$RHtrue0 = (20.64 - 25)(8 \times 1880 \times 10^{-5} + 10^{-2} + RHlin0 = 61.54 \tag{5}$$

$$RHlin0 = -2.8 \times 1880^2 10^{-6} + 4.05 \times 188010^{-2} - 4 = 62.24, \tag{6}$$

つまり, 摂氏 $T0 = 20.64$, 相対湿度 $RHtrue0 = 61.54$ となる.

17 付録:ソースコード

17.1 LD スクリプト

以下のファイルは LD スクリプトです:

```
/* Default linker script, for normal executables */
OUTPUT_FORMAT("coff-h8300")
OUTPUT_ARCH(h8300hn)
ENTRY("_start")
MEMORY
{
        /* 0xc4 is a magic entry.  We should have the linker just
           skip over it one day...  */
        vectors : o = 0x0000, l = 0xc4
        magicvectors : o = 0xc4, l = 0x3c
        ram    : o = 0x0100, l = 0xfdfc
        /* The stack starts at the top of main ram.  */
        topram : o = 0xfefc, l = 0x4
        /* At the very top of the address space is the 8-bit area.  */
        eight : o = 0xff00, l = 0x100
}
SECTIONS
{
.vectors :
        {
```

```
        SHORT (ABSOLUTE(_start))        /*  0 Reset vector */
          /* Use something like this to place a specific
             function's address into the vector table.
             SHORT (ABSOLUTE (_foobar)).  */
          *(.vectors)
        }  > vectors
.text :
        {
          *(.rodata)
          *(.text)
          *(.strings)
           _etext = . ;
        }  > ram
.tors :
        {
          ___ctors = . ;
          *(.ctors)
          ___ctors_end = . ;
          ___dtors = . ;
          *(.dtors)
          ___dtors_end = . ;
        } > ram
.data :
        {
          *(.data)
          *(.tiny)
           _edata = . ;
        }  > ram
.bss :
        {
           _bss_start = . ;
          *(.bss)
          *(COMMON)
           _end = . ;
        }  >ram
.stack :
        {
           _stack = . ;
          *(.stack)
        }  > topram
.eight :
        {
          *(.eight)
        }  > eight
.stab 0 (NOLOAD) :
        {
          [ .stab ]
        }
.stabstr 0 (NOLOAD) :
        {
          [ .stabstr ]
        }
}
```

17.2 スタートアップファイル

以下のファイルはスタートアップファイルです:

```
; h8/300 and h8/300h start up file.
```

```
        .h8300hn
        .section .text
        .global _start
_start:
        mov.w   #_stack,sp
        ;; copy DATA from ROM
        mov.w   #___dtors_end,r0
        mov.w   #_edata,r1
        mov.w   #___data,r2
.loop0: mov.w   @r0,r3
        mov.w   r3,@r2
        add.w   #2,r0
        add.w   #2,r2
        cmp.w   r1,r2
        blo     .loop0

        ;; clear BSS
        mov.w   #_edata,r0
        mov.w   #_end,r1
        mov.w   #0,r2
.loop:  mov.w   r2,@r0
        add.w   #2,r0
        cmp.w   r1,r0
        blo     .loop
        jsr     @_main

        .section .stack
_stack: .long   1
```

17.3 アプリケーションプログラム

以下のファイルはアプリケーションプログラムです:

```
        .h8300hn
        .equ    SMR3,   0xffa8
        .equ    BRR3,   0xffa9
        .equ    SCR3,   0xffaa
        .equ    TDR3,   0xffab
        .equ    SSR3,   0xffac
        .equ    PMR1,   0xffe0
        .equ    PDR8,   0xffdb
        .equ    PCR8,   0xffeb
        .equ    ADCSR,  0xffb8
        .equ    ADDRA,  0xffb0
        .equ    MSTCR1, 0xfff9
        .equ    SCR_TE,         5
        .equ    SSR_TDRE,       7
        .equ    SSR_TEND,       2
        .equ    PMR_TXD,        1
        .equ    ADCSR_ADF,      7
        .equ    ADCSR_ADST,     5
        .equ    PDR8_DO,        5
        .equ    PDR8_DI,        6
        .equ    PDR8_SCK,       7
        .equ    DI_FILTER,      0x40

        .section        .data
```

```
OUT:    .asciz  "0000 0000\n"

        .section        .text
        .global _main

_main:
        jsr     @init_port
        jsr     @init_serial
        jsr     @init_sht
infty_loop:
        jsr     @getThermo
        jsr     @transto_dec
        jsr     @getHumidity
        jsr     @transto_dec2
        jsr     @send_string
        jsr     @wait1s
        jmp     @infty_loop

init_port:
        push.w  r0
        mov.b   #0x00,  r0l
        mov.b   r0l,    @PDR8
        mov.b   #0xA0,  r0l
        mov.b   r0l,    @PCR8
        pop.w   r0
        rts

init_serial:
        push.w  r0
        mov.b   #0x00,  r0l
        mov.b   r0l,    @SCR3
        mov.b   r0l,    @SMR3

        mov.b   #51,    r0l
        mov.b   r0l,    @BRR3

        jsr     @wait4m

        pop.w   r0
        rts

init_sht:
        jsr     @wait1s
        rts

getHumidity:
        jsr     @startTransmission
        jsr     @requestHumidity
        jsr     @waitMeasurementFinish
        jsr     @receiveData
        rts

getThermo:
        jsr     @startTransmission
        jsr     @requestThermo
        jsr     @waitMeasurementFinish
        jsr     @receiveData
        rts

startTransmission:
```

```
        bset    #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_DO,       @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8

        bset    #PDR8_SCK,      @PDR8:8
        bset    #PDR8_DO,       @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        rts

requestHumidity:
        mov.b   #0x05,  r0l
        jsr     @putByte

        jsr     @receiveAck
        rts

requestThermo:
        mov.b   #0x03,  r0l
        jsr     @putByte

        jsr     @receiveAck
        rts

waitMeasurementFinish:
        bset    #PDR8_DO,       @PDR8:8
        jsr     @wait80u
measuring:
        jsr     @wait4m
        btst    #PDR8_DI,       @PDR8:8
        bne     measuring
        rts

receiveData:
        push.l  er1
        mov.l   #0,     er0
        mov.w   #0,     r1
        mov.w   #7,     e1

        jsr     @getByte
        mov.b   r0l,    r0h
        jsr     @sendAck

        jsr     @getByte
        mov.b   r0l,    r0l
        jsr     @skipAck

        pop.l   er1
        rts

getByte:
        mov.b   #0,     r0l

        bset    #PDR8_DO,       @PDR8:8
        jsr     @wait80u

        bset    #PDR8_SCK,      @PDR8:8
        bld     #PDR8_DI,       @PDR8:8
        bst     #7,     r0l
        bclr    #PDR8_SCK,      @PDR8:8
```

```
        bset    #PDR8_SCK,      @PDR8:8
        bld     #PDR8_DI,       @PDR8:8
        bst     #6,     r0l
        bclr    #PDR8_SCK,      @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bld     #PDR8_DI,       @PDR8:8
        bst     #5,     r0l
        bclr    #PDR8_SCK,      @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bld     #PDR8_DI,       @PDR8:8
        bst     #4,     r0l
        bclr    #PDR8_SCK,      @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bld     #PDR8_DI,       @PDR8:8
        bst     #3,     r0l
        bclr    #PDR8_SCK,      @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bld     #PDR8_DI,       @PDR8:8
        bst     #2,     r0l
        bclr    #PDR8_SCK,      @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bld     #PDR8_DI,       @PDR8:8
        bst     #1,     r0l
        bclr    #PDR8_SCK,      @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bld     #PDR8_DI,       @PDR8:8
        bst     #0,     r0l
        bclr    #PDR8_SCK,      @PDR8:8

        rts

putByte:
        bld     #7,     r0l
        bst     #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        bld     #6,     r0l
        bst     #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        bld     #5,     r0l
        bst     #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        bld     #4,     r0l
        bst     #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        bld     #3,     r0l
        bst     #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        bld     #2,     r0l
        bst     #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        bld     #1,     r0l
        bst     #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
```

```
        bld     #0,     r0l
        bst     #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8

        rts

receiveAck:
        bset    #PDR8_DO,       @PDR8:8
        jsr     @wait80u

        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        rts

sendAck:
        bclr    #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        rts

skipAck:
        bset    #PDR8_DO,       @PDR8:8
        bset    #PDR8_SCK,      @PDR8:8
        bclr    #PDR8_SCK,      @PDR8:8
        rts

conv_ad:
        extu.l  er0
        bset    #ADCSR_ADST,    @ADCSR:8
ad_loop:
        btst    #ADCSR_ADF,     @ADCSR:8
        beq     ad_loop
        mov.w   @ADDRA, r0
        shlr.w  r0
        shlr.w  r0
        shlr.w  r0
        shlr.w  r0
        shlr.w  r0
        shlr.w  r0
        rts

transto_dec:
        push.l  er0
        push.l  er1
        push.l  er2
        mov.w   #10,    e1
        mov.w   #OUT+4, r2
        mov.w   #4,     e2
dec_loop:
        jsr     @div_dec
        dec.w   #1,     e2
        bne     dec_loop
        pop.l   er2
        pop.l   er1
        pop.l   er0
        rts

transto_dec2:
        push.l  er0
```

```
        push.l  er1
        push.l  er2
        mov.w   #10,    e1
        mov.w   #OUT+9, r2
        mov.w   #4,     e2
dec2_loop:
        jsr     @div_dec
        dec.w   #1,     e2
        bne     dec2_loop
        pop.l   er2
        pop.l   er1
        pop.l   er0
        rts

div_dec:
        divxu.w e1,     er0
        mov.w   e0,     r1
        extu.l  er0
        add.b   #0x30,  r1l
        mov.b   r1l,    @-r2
        rts

send_string:
        push.l  er0
        push.l  er5
        bset    #SCR_TE,        @SCR3:8
        bset    #PMR_TXD,       @PMR1:8
        mov.w   #OUT,   r5
put_ch:
        btst    #SSR_TDRE,      @SSR3:8 ; if SSR_TDRE=0, Z=1
        beq     put_ch
        mov.b   @r5+,   r0l
        cmp.b   #0x00,  r0l
        beq     send_fin
        mov.b   r0l,    @TDR3
        bclr    #SSR_TDRE,      @SSR3:8
        jmp     @put_ch
send_fin:
        btst    #SSR_TEND,      @SSR3:8
        beq     send_fin
        bclr    #SCR_TE,        @SCR3:8
        bclr    #PMR_TXD,       @PMR1:8

        pop.l   er5
        pop.l   er0
        rts

wait1s:
        mov.l   #0x186a00,      er6
        jmp     @time11
wait4m:
        mov.l   #0x1900,        er6
        jmp     @time11
wait80u:
        mov.l   #0x80,  er6
time11:
        sub.l   #1,     er6
        bne     time11
        rts
```

```
        .end
```

参考文献

[1] Runesas *H8/3664* リファレンスマニュアル,
http://www.renesas.com/jpn/products/mpumcu/16bit/h8300h/

[2] Muneyuki IWATA *Linux* で *H8* のフリー開発環境を*!*,
http://iwatam-server.dyndns.org/hardware/h8comm/doc/CrossDevel-jp.html/

[3] 白土 義男 *H8* ビギナーズガイド, 東京電機大学出版局, 2000

第 VII 部
Pascal プログラミングの基礎 第一回

(著者) J 科 egashira

18 プログラムを書いてみる

下を見るといきなりプログラムがありますが、決して難しいことを行なっているわけではありません。一般の参考書などでは初めに出入力の方法と基本的な算術演算子の扱い方を書くと思うのですが、それだと読者の方に面白みが無いと思いますし、こちらもページの都合があるので見れば分かるところはさらっと流してより実用的な部分に力をいれてプログラム作りの面白さを知って頂きたいと思います。例題 1 では、出入力及び算術演算子、算術関数の扱い方を知ってもらいます。

(例題 1) 三辺の長さが a,b,c であることが分かっている三角形の面積 area は、$s = \frac{a+b+c}{2}$ とすると、ヘロンの公式より

$$area = \sqrt{s(s-a)(s-b)(s-c)} \tag{7}$$

となります。この公式を用いて、三辺の長さを入力したら、三角形の面積を出力するプログラムを作りなさい。

〈 **解答例** 〉

```
 1:    program Heron(input,output);
 2:    var a, b, c, s, area : real;
 3:    begin
 4:      writeln(output,'three reals ?');
 5:      write(output,' a = ');
 6:      readln(input,a);
 7:      write(output,' b = ');
 8:      readln(input,b);
 9:      write(output,' c = ');
10:      readln(input,c);
11:      s:= (a+b+c)/2;
12:      area:= sqrt(s*(s-a)*(s-b)*(s-c));
13:      writeln(output,'三角形の面積 area は ',area:8:4,'です。')
14:    end.
```

```
sigma[68]%  ./a.out
three reals ?
 a = 2.3
 b = 3.4
 c = 4.5
三角形の面積 area は  3.8165 です。
```

図 1： 例題 1 の実行結果

では、解説に入りましょう。まず、1 行目の program に続く “名前” はプログラム全体を表す任意の名前です。英数字ならば何でも結構ですがハイフンは何故かエラーになりますから、入れないようにしましょう。またその直後にある括弧内には例題にあるように通常 input,output と書きます。また、その後には ‘;’ をつけるのを忘れないで下さい。

次に、2 行目の var はその後に主プログラムで使う変数を宣言します。変数の名前には英数字を一文字以上にして割り当てることが出来ますが、数字を変数の先頭または単独で使うことは出来ません。のように値を入れます。そして、‘:’ で区切った後にその変数の型を書きます。この場合 real というのは宣言した変数にどのような値が入れられるかを指定しています。real の他にも基本的なデータ型として以下に挙げる 3 つがあります。

標準のデータ型	型の名前	型の持ち得る値の例
integer	整数型	...,-2,-1,0,1,2...
real	実数型	0.5,12.3,-4.2,3.62e-03
Boolean	論理型	false,true(2 つだけ)
char	文字型	a,V,%

表 2: データ型の種類

まず、プログラムの主体に入る前に注意点をいくつか。3 行目と 10 行目にそれぞれ begin と end と書かれています。プログラムの主体は必ずこの 2 つの間に無ければいけません。また、end. 直前の最後の行以外は文が一つ終わる度に ‘;’ で区切ります。これを抜かすとエラーになるので注意して下さい。さて、やっとプログラムの主体に入ります。まずは、小ネタで行きますが、4 行目と 5 行目とでは微妙に違うのですが、どこか分かりますか?そう、write の後に ln があるかないかです。この ln を付けとその作業が終わったときに改行するようになります。以下 10 行目まで同じことを行なっています。また、文を出力したい場合には’—文—’ と書きます。但し、変数に値が入っているときにいれる必要はなく’—文—’, 変数,’—文—’ と書き文と変数をカンマで区切ります。

11 行目からは算術演算子がでて来ているので、説明するよりも一覧表 (表 2) 作ったのでそちらを参考にして下さい。

12 行目に注意点があるとすれば、‘:=’ と ‘sqrt’ についてでしょう。変数に値を代入するときにはただ ‘=’ でつなげばよいというわけではなく、左に変数を、右に値を持つものを持って来て ‘:=’ でつながなければいけません。

次に、sqrt というのは算術関数と呼ばれるものの一つで、sqrt(x) と書いて $\sqrt{x}$ という演算結果がでます。sqrt 以外にも表 3 にあるようなものがあります。

+	整数型、実数型の足し算を行なう
-	整数型、実数型の引き算を行なう
*	整数型、実数型の掛け算を行なう
/	実数型の割算を行なう
mod	割算の余りを求める

表 3: 算術演算子一覧

13 行目が終われば Pascal プログラミングで必須と思われることは大体終わりです。ここで覚えて欲しいことは一つで area の後に ‘:8:4’ とあります。これの用途についてです。その部分が無い場合の実行結果を挙げるので比較して下さい。

見れば分かるように、変数 area を出力すると小数点以下を延々と書いていき最後に e+00 とわけの分からないものまでついてきます。一般的には ‘:w:f’ と書いて w 文字分の全体幅の中に小数点以下を f 文字分とって右揃えで出力するといった意味にるのでそれを除いたためです。そして、e+00 というのは、10 の 0 乗を意味し、e+0n と書かれていた場合、10 の n 乗を意味します。

19 制御文

プログラムを書く上で何が何でも習得したいのがこの制御文でしょう。この制御文が扱えるようになるとプログラムの幅が飛躍的に広がりまた、自分の作りたいプログラムがある程度書けるようになります。そうすると、

関数名または手続き名	multicolumn1—c—機能	関数の場合は型
abs(x)	引数の絶対値を返す	integer か real
arctan(x)	引数の逆正接値を返す	real
cos(x)	引数の余弦値を返す	real
exp(x)	引数の指数関数値を返す	real
frac(x)	引数の小数部分を返す	real
int(x)	引数の整数部分を返す	real
ln(x)	引数の自然対数値を返す	real
sin(x)	引数の正弦値を返す	real
sqr(x)	引数の平方根を返す	integer か real
sqrt(x)	引数の平方根を返す	real

表 4: 算術関数一覧

```
sigma[3]%  ./a.out
three reals ?
 a = 2.3
 b = 3.4
 c = 4.5
三角形の面積 area は 3.816490534509418e+00 です。
```

表 5: 実行結果 (2)

プログラミングが楽しくなって来ますから、この機会に覚えてしまって下さい。

19.1 if 文

では、いきなりですが例題から入りましょう。

(例題 2) 2 つの実数を読み込み、それらの最大値と最小値を出力するプログラムを作りなさい。但し、二つが等しいときにはそれを知らせるようなものにすること。

〈 **解答例** 〉

```
 1:  program minMax(input,output);
 2:  var a, b, min, max : integer;
 3:  begin
 4:    writeln(output,'二つの整数を入力して下さい');
 5:    write(output,'a = ); readln(input,a);
 6:    write(output,'b = ); readln(input,b);
 7:    if a < b then begin min := a; max := b end
 8:    else if a > b then begin min := b; max := a end
 9:    else writeln(output,'最大値と最小値はなく二つの整数は等しい');
10:    if (a < b) or (a > b) then
11:       writeln(output,'min = ',min : 1,' max = ',max : 1)
12:  end.
```

```
sigma[80]%  ./a.out
二つの整数を入力して下さい
a = 12
b = 10
min = 10 max = 12
sigma[81]%  ./a.out
二つの整数を入力して下さい
a = 1
b = 1
最大値と最小値はなく二つの整数は等しい
```

表 6: if 文 実行結果

制御文の筆頭とも言える if 文です。今回の Pascal プログラミングでは著者の時間とページの都合で if 文と repeat 文しかお教えできませんが、この二つを覚えておくだけでもプログラミングの自由度をかなり高くできると思います。

では、解説に入りましょう。6 行目までは問題ないと思うので、7 行目の if 文の説明から入りたいと思います。まず、if 文の書き方から入ります。if 文には代表的なパターンが 3 つありそれらは以下の通りです。

(a) if 論理式 then 文

(b) if 論理式 then 文 1
else 文 2

(c) if 論理式 1 then 文 1
else if 論理式 2 then 文 2
:
:
else if 論理式 (n-1) 文 (n-1)
else 文 n

if 文の説明にはいる前にまたいくつか説明を要するものがあります。その一つ目が、論理式についてです。論理式と言うのは以下の 3 つによって定義されます。

1. 論理型の定数、及び変数は論理式
2. a と b を整数型、実数型、文字型のいずれかの型の式 (ただし、a、b はいず れも同じ型、または一方が整数型で他方が実数型) としたとき、$a = b,\ a <> b\ a < b\ a <= b\ a > b\ a >= b$ はいずれも論理式
3. p と q を論理式としたとき、not(p)、(p)and(q)、(p)or(q) も論理式

ここで、=, <>, <, <=, >, >= を関係演算子と言います。それらの定義を表に示します。数学で出てくるものがほとんどなので覚える程ではないかも知れません。また、not、and、or を論理演算子と呼びます。これらはそのままの意味なので表は割愛させて頂きます。

以上を踏まえると、

- (a) は、論理式が真のときに文を実行し、偽のときには何も行なわない
- (b) は、論理式が真のときに文 1、を偽のときには文 2 を実行する
- (c) は、論理式 1、論理式 2、.... を順々に真になるまで調べていき、初めて真になる論理式が論理式 i ならば、文 i を実行しこの if 文全体を終え全ての論理式が偽のときは、文 n を実行します

関係演算子	定義
a = b	a と b が等しければ true、そうでなければ false
a 〈 〉 b	a と b が等しくなければ true、そうでなければ false
a 〈 b	a が b より小さければ true、そうでなければ false
a 〈= b	a が b 以下であれば true、そうでなければ false
a 〉 b	a が b より大きければ true、そうでなければ false
a 〉= b	a が b 以上であれば true、そうでなければ false

表 7: 関係演算子の定義

となります。

ちょっとだけ注意して頂きたいのは if 文というのは (a) の場合を除けば、else 文が終わって初めて if 文全体が終わったことになるので、';' をつける場合は、else 文が終わった後にしないといけません。

さて、例題の解説に戻りたいと思います。といっても、以上のことが分かっていれば例題で分からない点はもうないと思います。ただ、2 つだけ注意して欲しいところがあります。1 つは些細なことなのですが、and や or で変数同士やその式を結ぶときには (変数の式)and(変数の) と言うように括弧でくり、not の場合も not(変数の式) として下さい。2 つ目は、if 文も文の一つなので、then や else の後にまた if 文が続くこともできます。実際 (c) のパターンは、else の後に if 文が続いている例です。

特に注意が必要なのは次の場合です。

if 論理式 1 then if 論理式 2 then 文 1 else 文 2

は、

if 論理式 1 then begin if 論理式 2 then 文 1 else 文 2 end

と解釈され、else は未対応の直前の if(または then) に対応すると定められています。そのため、もし論理式 1 が偽のときに文 2 を実行したいのなら、

if 論理式 1 then begin if 論理式 2 then 文 1 end else 文 2

と書かなくてはいけません。

19.2 repeat 文

続いては、repeat 文に入ります。

(例題 3) 正整数 S を読み込み、初めて $\sum_{k=1}^{n} 1/k \geq$ S となる n を求めるプログラムを作りなさい。

〈 **解答例** 〉

```
1:  program sum(input,output);
2:  var b, d : integer;
     a, c, e : real;
3:  begin
4:     writeln(output,'正の整数を入力して下さい');
5:     repeat
6:        write(output,'S = '); readln(input,a);
7:        if (a <> (trunc(abs(a))))
```

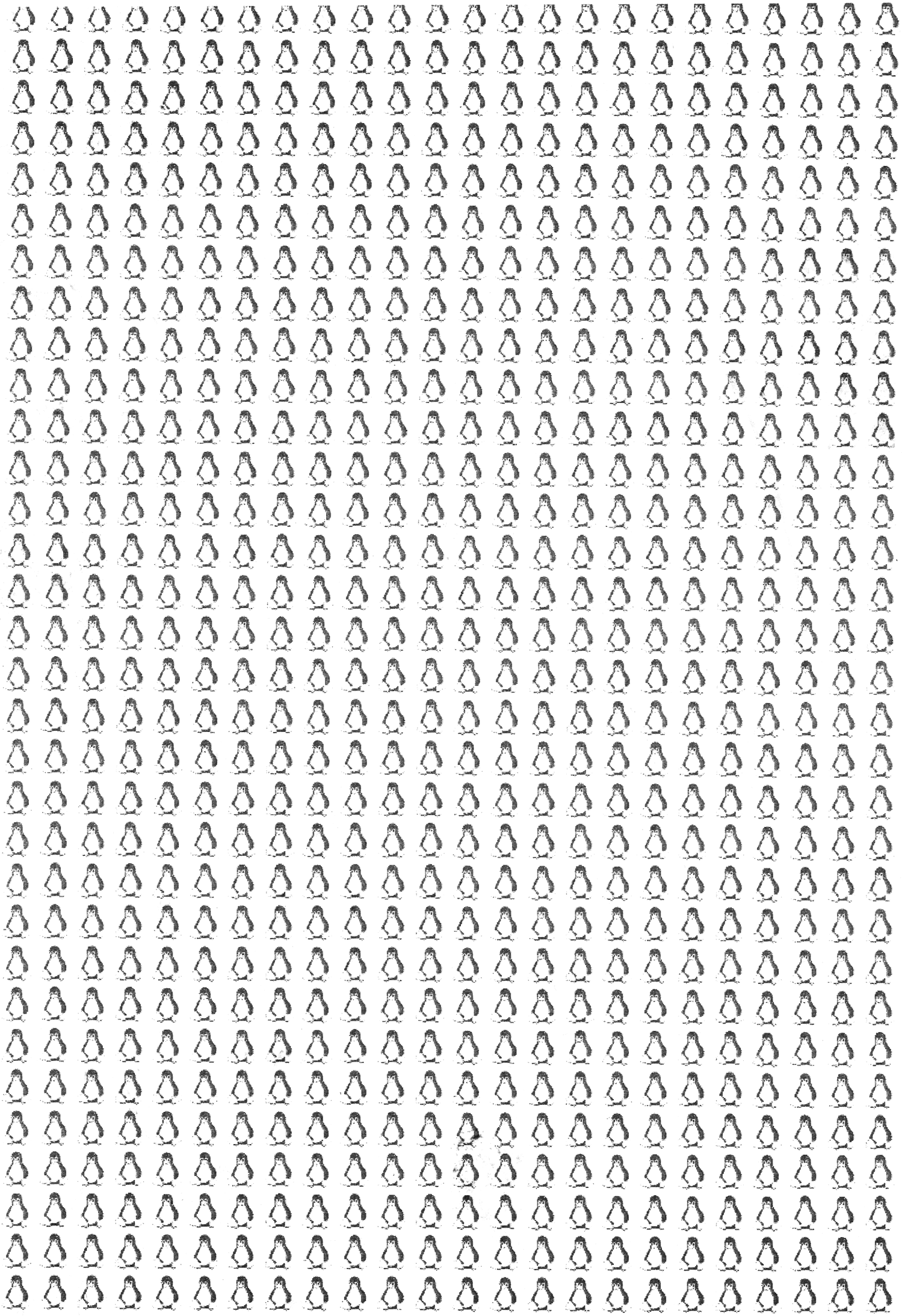
```
 8:           then writeln(output,'S には正の整数を入力して下さい');
 9:     until a = (trunc(abs(a)));
10:     b := 1;
11:     c := 0;
12:     d := 0;
13:     e := 0;
14:     repeat
15:        d := (1 + d);
16:        e := (1/d);
17:        c := (c + e);
18:     until c >= a;
19:     writeln(output,'1+1/2+1/3+...+1/n が S 以下になるのは n = ',d,' の時
です。')
20:  end.
```

7 行目に trunc という見慣れない演算子らしきものがあると思います。これは、変換関数の一つで、括弧内の値の整数部分を返すものです。他には repeat 文以外で特筆しなければいけないものはないので repeat 文の構造の説明をしたいと思います。

repeat 文 1; 文 2; ..... ; 文 n until 論理式

repeat 文は、文 1〜文 n を、論理式の値が true になるまで、繰り返し実行します。各繰り返しの最後に論理式の値が計算され、true であれば、repeat 文全体の実行が終了し、false であれば、繰り返しの先頭 (文 1) に戻ります。また、until の後には、

until (条件文 1) (and または or) (条件文 2) ...

というように複数の条件文を and や or でつなげることができます。

以上で今回の 'Pascal プログラミングの基礎' は終わりです。ここに記述されているものだけではまだ十分なプログラムは書けないでしょうが、その辺りはご容赦下さい。