岡山大学経済学会雑誌31(3),1999, 255～264

LZH の中での末尾数字を意識したファイル・ソート

藤　本　喬　雄

ファイルを次々，LZH ファイル（吉崎栄寿氏の LHA（ver. 2 .13：1991年）によって圧縮されたファイル）の中へ追加保存していくと，時折ソートしたくなる。各月ごとの LZH ファイルを構成する場合，各週ごとに作成されたファイルが追加される。その際，ファイル名でソートしたいのだが，普通のソート・プログラムは，各文字ごとに比較していくので，昇順の場合，“AAA10. DOC”の方が“AAA2. DOC”より前にきてしまう。末尾の数字を全体として数とみなし，後者を前におきたいので，プログラムを作成した。使用した言語は，1985年に Borland（現在の Inprise）より発売された，Turbo Pascal ver. 3 である。DOS 上で汎用的で，Windows でも，DOS に抜ければ使用できる。メモリが，64KB，128KB の時代なので，データ型として，Word や Longint がない。以下，プログラムの要点を説明する。

(1) ファイル名リストには，レコード型（FileContent）の配列を用いた。ファイル名と，LZH ファイルの中での，開始バイトと長さを含む。この配列は，2000個のファイルを含むことができる。個々のファイル用の動的なメモリ配分はやめて，最初から，ヒープ・エリアに2000個分のメモリ40KB をとった。現在の普通のメモリ量なら数万個を扱えるが，2000個で充分であろう。

(2) 元のファイルのタイム・スタンプを変えないために，INT 21 h（AH＝57 h）を使って，DOS のファンクション・コールを行った。Turbo

Pascal ver. 4 の，SetFTime と異なり，ファイルを閉じてからやらないといけない。

(3) ソートは，quick sort を使った。C言語の quicksort 関数がないので，BASIC 用のものを，Pascal に書き直した。

(4) ファイルの読み書きは，BlockRead，BlockWrite を用いた。おもしろいことに，長さのパラメータは整数としては負数でも，Word として解釈してくれるのである。今回は，このことは利用しなかった。ファイルの1レコードは，1バイトとして設定しオープンした。

(5) LZH の中で使われている，Longint に対処するため real 型の変数を用いた。ファイルの長さにも，real 型を使った。procedure sort の中で，Longint を1バイトづつ，real に変換している。

(6) 次のファイル情報を見つけるために，48バイト先読みを行った。BlockRead を行う前に，LongSeek で，適切な切れ目に戻す必要が生じた。

(7) ファイル名の末尾の最大3桁（半角）を数とみなすことにした。末尾からスキャンして，数字でないものが現れたら打ち切る。打ち切られた前の部分が共通であれば，数としての大きさが順番を決める。procedure CompString の部分である。

(8) 付録のプログラムには，デバッグ中に使ったメッセージ印刷をコメントとして残してある。

LZH ファイルにおける関連した部分の構造について，述べておこう。圧縮された1個1個のファイルは，LZH ファイルの中で独立したバラバラの状態である。目次のようなものはない。3バイト目に，“－lh”（2Dh, 6Ch, 68h）がある。8バイト目から4バイトが，ファイル・ディレクトリからとったファイルの長さである（procedure sort の真ん中あたりの処理を参照)。15バイト目から2バイトがファイル更新の時刻，17バイト目から2バイトが年月日を示す。22バイト目にファイル名の長さが1バイトで表わされ

ている。圧縮されたファイル全体の長さは，ファイルの長さ＋ファイル名の長さ＋27，となる。付録プログラムにおける，procedure sort の r_length の計算を見られたい。Longint がないので，この変数は real 型にしている。現在の開始点に，この r_length を加えれば次のファイルの開始点になる。

プログラム自体のテキスト・ファイルは，10,380バイト，コンパイル後の COM ファイルが，15,175バイト，EXE に変換して，LZEXE（Fabrice Bellard 氏によるもの：1989年）で圧縮すると，11,595バイトになった。

LHA は，ver. 2 .13であることに注意してほしい。長さ 1MB 以下のファイルなら１，２秒でソートは終わるが，時間が異様にかかる場合，ソート後の長さが異なる場合，全て順調に修了したように見えてもソート後のファイルに対して LHA がファイル異常を報告する場合，元の LZH ファイルが，ver. 2 .13より旧い版で作成されたと考えてよい。もっとも流布したこの版で解凍，圧縮をし直せばよい。1991年より旧い LZH ファイルの場合，この可能性が高い。

種々の LZH ファイルのソートをしてみたが，27MB のもので，1900個以上のファイルを含む場合，Intel80486，66MHz のＰＣにおいて，ソートに３分かかった。解凍して DOS 上でソートし，マニュアルで入れ替えるよりもましであると考える。

Windows もファイル・リストのオプションで，この種のソートを許すようにすべきであろう。IE4，IE5 のスクリプトが既に利用可能かも知れない。

付録　Pascal（ver. 3）プログラム

```
program LZHSORT;(* Sorting LHZ archives for Turbo Pacal ver.3.01A *)
                (* TF:1999-8-24(Tue.), 8-29(Sun):Yashima: Cloudy. *)
                (* NB: trailing numerals up to 3 digits. *)
const
   MAX_FILE = 2000; (* Max number of files allowed in a LZH *)
   buf_len = 24576;
   r_buf_len = 24576.0;
   buf_lena= 24624; (* to read a little more additional bytes *)
type RegRec=record
    case Integer of
     1: (AX,BX,CX,DX,BP,SI,DI,DS,ES,Flags: integer);
     2: (Al,AH,BL,BH,CL,CH,DL,DH: byte);
  end;
type FileContent=record
     Name : string[12];
     Start: real;
     Length : real;
  end;
type
   str12 = string[12];
   str13 = string[13];
type
   FileArray = array[1..MAX_FILE] of FileContent;
var
   PFileArray : ^FileArray;
   FFI,OFI,EXT :str13;
   p_str12 : ^str12;
   p_byte_l : ^byte;
   p_byte_nl : ^byte;
   H_FFI, H_OFI: integer; (* handles *)
   f1, f2  :file;
   lastbyte:real;
   i,j     :integer;
   NumRead, NumWritten : integer;
   buf     :array[1..buf_lena] of byte;
   s_buf   :integer;
   Regs    :RegRec;
   f_date, f_time: integer;

procedure initialize;
begin
  if (ParamStr(1)<>'')
    then
      FFI:=ParamStr(1)
    else begin
      write('*** Please key in the Input-file name.(ext.default .LZH). = ');
      readln(FFI);
    end;
  if (ParamStr(2)<>'')
    then
      OFI:=ParamStr(2)
    else begin
      write('*** Now key in the Output-file name.  (ext.default .LZ2). = ');
      readln(OFI);
    end;

  if POS('.',FFI)=0
    then FFI:=FFI+'.LZH';
  if OFI=''
    then begin
```

```
    OFI:=FFI;
    OFI[0]:=CHR(POS('.',FFI)-1);
  end;
if POS('.',OFI)=0 then OFI:=OFI+'.LZ2';

for i:=1 to Length(FFI) do FFI[i]:=UpCase(FFI[i]);
for i:=1 to Length(OFI) do OFI[i]:=UpCase(OFI[i]);

if FFI=OFI then
 begin
    writeln('xxx The same name for Input & Output Files! Try again.');
    halt(1);
 end;

i := POS('.',FFI);
if i>1 then EXT:=Copy(FFI, i+1, 3);
if (i=0) or (EXT<>'LZH') then
begin
  writeln('xxx Input File is not "LZH"! Try again.');
  halt(1);
end;

      (* to make ASCIIZ strings *)
 i:=Length(FFI);
 Insert(Chr(0),FFI,i+1);
 FFI[0]:=Chr(i);   (* Insert makes the string longer. *)
 i:=Length(OFI);
 Insert(Chr(0),OFI,i+1);
 OFI[0]:=Chr(i);

 assign(f1,FFI);
 assign(f2,OFI);
  {$I-}
 Reset(f1,1);
  {$I+}
 if IOresult<>0
   then begin
     writeln('xxx Input File does not exist! Try again.');
     halt(1);
   end;

 Rewrite(f2,1);

 lastbyte:=LongFileSize(f1)+0.001;
 writeln('... File Length of ',FFI,' = ', lastbyte:10:0,' bytes.');

      (* s_buf: a negative value OK. See Blockread. *)
 s_buf:=SizeOf(buf);

end; (* end of initialize *)

procedure open_handle(var n_file: str13; var h_file: integer);
begin
 with Regs do
 begin
  AX:=$3D00;
  DS:=DSeg;
  DX:=Ofs(n_file)+1;
  MsDos(Regs);
  h_file:=AX;
 end;
end;
```

```
procedure close_handle(h_file:integer);
begin
 with Regs do
 begin
  AX:=$3E00;
  BX:=h_file;
  MsDos(Regs);
 end;
end;

procedure get_time(var n_file: str13; var h_file,
                                        f_date, f_time: integer);
begin
  open_handle(n_file, h_file);

 with Regs do
 begin
  AX:=$5700;    (* Get the time stamp of Input file. *)
  BX:=h_file;
  MsDos(Regs);
  f_date:=DX;
  f_time:=CX;
 end;

 close_handle(h_file);
end; (* end of get_time *)

procedure set_time(var n_file: str13; var h_file: integer;
                                        f_date, f_time: integer);
begin
  open_handle(n_file, h_file);

 with Regs do
 begin
  AX:=$5701;    (* Set the time stamp of Input file. *)
  BX:=h_file;    (* AL=$01 *)
  CX:=f_time;
  DX:=f_date;
  MsDos(Regs);
 end;

  close_handle(h_file);
end; (* end of set_time *)

function CompString(S1, S2: str12): Boolean;
var
 S10, S20 : str12;
 p1, p2: integer;
 n1, n2 : integer;
 n11, n12, n13, n21, n22, n23: integer;
 ie : integer;
begin
 (* writeln('S1=',S1,'; S2=',S2); *)
 S10:=S1; S20:=S2;
 p1:=Pos('.', S1)-1;
 p2:=Pos('.', S2)-1;
 if p1>0 then S1[0]:=Chr(p1);
 if p2>0 then S2[0]:=Chr(p2);

 n11:=0; n12:=0; n13:=0;
 if (S1[p1]>='0') and (S1[p1]<='9') then
```

```
begin
    Val(S1[p1], n11, ie);
    S1[0]:=Chr(p1-1);
  if (S1[p1-1]>='0') and (S1[p1-1]<='9') then
  begin
     Val(S1[p1-1], n12, ie);
     S1[0]:=Chr(p1-2);
   if (S1[p1-2]>='0') and (S1[p1-2]<='9') then
    begin
       Val(S1[p1-2], n13, ie);
       S1[0]:=Chr(p1-3);
    end;
  end;
end;
n1:=n13*100+n12*10+n11;

n21:=0; n22:=0; n23:=0;
if (S2[p2]>='0') and (S2[p2]<='9') then
begin
   Val(S2[p2], n21, ie);
   S2[0]:=Chr(p2-1);
 if (S2[p2-1]>='0') and (S2[p2-1]<='9') then
  begin
    Val(S2[p2-1], n22, ie);
    S2[0]:=Chr(p2-2);
   if (S2[p2-2]>='0') and (S2[p2-2]<='9') then
    begin
       Val(S2[p2-2], n23, ie);
       S2[0]:=Chr(p2-3);
    end;
  end;
end;
n2:=n23*100+n22*10+n21;

(* writeln('...', S1,', ' , S2, '; n1:=',n1,';n2=',n2); *)

 if S1<S2 then CompString:=TRUE;
 if S1=S2 then if n1<n2 then CompString:=TRUE
    else
    if n1=n2 then
    begin
     if S10<S20 then CompString:=TRUE
     else CompString:=FALSE;
    end
    else CompString:=FALSE;
 if S1>S2 then CompString:=FALSE;
end;

function CompStringE(S1, S2: str12): Boolean;
begin
 if CompString(S1, S2) then CompStringE:=TRUE
 else
   if S1=S2 then CompStringE:=TRUE
   else CompStringE:=FALSE;
end;

procedure SwapArray(A,B:integer);
var
   Temp: FileContent;
begin
     Temp:=PFileArray^[A];
     PFileArray^[A]:=PFileArray^[B];
```

```
    PFileArray^[B]:=Temp;
end;

procedure QuickSort(Low, High: integer);
var
   RandIndex : integer;
   Partition : str12;
   i,j : integer;
begin
   if Low < High then
   begin

     (* the case of 2 elements *)
      IF High - Low = 1 then begin
       IF CompString(PFileArray^[High].Name, PFileArray^[Low].Name) then
          SwapArray(Low,High);
      end
      else
      begin

         (* centre element randomly chosen placed at the end. *)
         RandIndex := RANDOM(High-Low+1)+Low;
          SwapArray(High,RandIndex);
         Partition := PFileArray^[High].Name;
         repeat
            (* compare the both ends toward the centre. *)
            i := Low;
            j := High;

        while (i < j) and (CompStringE(PFileArray^[i].Name, Partition)) do
              i := i + 1;
        while (j > i) and (CompStringE(Partition, PFileArray^[j].Name)) do
              j := j - 1;

            (* if not reached the centre, the two elements *)
            (* are exchanged.                              *)
            if i < j then
              begin
                SwapArray(i,j);
              end;
         until i >= j;

         (* the centre element at the right place. *)
          SwapArray(i,High);

         (* QuickSort is called recursively. *)
         (* to suppress the use of stack     *)
         (* the lesser elements done first.  *)
         if (i - Low) < (High - i) then
          begin
            QuickSort(Low, i - 1);
            QuickSort(i + 1, High);
          end
          else
          begin
            QuickSort(i + 1, High);
            QuickSort(Low, i - 1);
          end;
      end; { if High-Low=1...else begin }
   end; { if Low<High }
end;
```

```
procedure sort;
var
 r_start, r_length: real;
 tmp_length: real;
 buf_count : integer;
 remainder, k : integer;
 i_start: integer;
begin
  r_start := 0.0;
  i := 1;
  j := 0;
  repeat
       (* BlockRead: integer treated as word. *)
    repeat
     BlockRead(f1, buf, s_buf, NumRead);
     j := j+1;
     LongSeek(f1, r_buf_len*j); (* Do not forget to adjust! *)
    until (r_start <= r_buf_len*j);

    if (NumRead <> 0) then
    begin
  (*     writeln('Numread=',NumRead); *)
      repeat
  (*    writeln('i=',i,'; j=',j,'; r_start=', r_start); *)
       i_start := Round(r_start - r_buf_len*(j-1) + 0.001);

        (* *** longint not available in TP v.3 *** *)
       tmp_length:=0.0;
       p_byte_1 := ADDR(buf[i_start+8]);
       tmp_length:=tmp_length + p_byte_1^;
       p_byte_1 := ADDR(buf[i_start+9]);
       tmp_length:=tmp_length + 256.0*p_byte_1^;
       p_byte_1 := ADDR(buf[i_start+10]);
       tmp_length:=tmp_length + 256.0*256.0*p_byte_1^;
       p_byte_1 := ADDR(buf[i_start+11]);
       tmp_length:=tmp_length + 256.0*256.0*256.0*p_byte_1^;

       p_str12 := ADDR(buf[i_start+22]);
       p_byte_n1:= ADDR(buf[i_start+22]);
       PFileArray^[i].Name   := p_str12^;
       PFileArray^[i].Start  := r_start;
       r_length := 27.0 + tmp_length + p_byte_n1^; (* LZH structure *)
       PFileArray^[i].Length := r_length;
       r_start := r_start + r_length;
       i := i + 1;
   (*    writeln(p_str12^, '; length=',r_length:10:0); *)
     until(r_start > (r_buf_len*j)) or (r_start > (lastbyte-5.0));
    LongSeek(f1, r_buf_len*j);
    end;
  until (NumRead = 0) or (r_start > (lastbyte-5.0));

  QuickSort(1,i-1);

  for j:=1 to (i-1) do begin
   LongSeek(f1, PFileArray^[j].Start);
    (* how many times *)
    buf_count := Trunc((PFileArray^[j].Length + 0.01)/r_buf_len);
 (*  writeln('buf_count=',buf_count); *)
   if (buf_count = 0) then
   begin
    BlockRead(f1, buf, Round(PFileArray^[j].Length+0.01), NumRead);
    BlockWrite(f2, buf, NumRead, NumWritten);
```

```
  end
  else
  begin
   for k:=1 to buf_count do
   begin
    BlockRead(f1, buf, buf_len, NumRead);
    BlockWrite(f2, buf, NumRead, NumWritten);
   end;
 remainder := Round(PFileArray^[j].Length - r_buf_len*buf_count + 0.01);
(*  writeln('remainder=',remainder); *)
   BlockRead(f1, buf, remainder, NumRead);
   BlockWrite(f2, buf, NumRead, NumWritten);
  end;
 end;

  buf[1]:=0;  (* the last byte of LZH *)
  BlockWrite(f2, buf, 1, NumWritten);

end; (* end of sort *)

procedure epilogue;
begin
 close(f1);
 close(f2);
  writeln('*** Sorting completed ! ***TF***');
end;

     (*** main routine ***)
begin
 New(PFileArray);
  initialize;
  get_time(FFI, H_FFI, f_date, f_time);
   sort;
  (*  for j:=1 to (i-1) do
    writeln(PFileArray^[j].Name); *)
  epilogue;
  set_time(OFI, H_OFI, f_date, f_time);
 Dispose(PFileArray);
end.
```