ベーシックライクコマンドの進め

本郷昭三

平成17年 3月 2日

目 次

1　はじめに

UNIX には、awk や sed 等強力な文字列操作コマンドがあるため、これを用いると、ほとんどの目的は達してしまう。 しかしながら、sed や awk はその多機能さゆえに、プログラムを組むたびに、マニュアルを参照せざるをえないこともしばしばある。 また、プログラムを読むときも、やや難解である。簡単にプログラムが作れまた読みやすいプログラムを作ることを目的としてベーシック風のコマンドを作成した。

2　文字列操作コマンド

2.1　「word」語の切り出し

- 区切り記号で文字列を分け、指定した番号の文字列を返す。
  複数番号を指定する場合はカンマ (,) で区切る
  例： word ”ab cd ef” 2,1 .....cd ab を返す
- 番号指定が無い場合は、分割数を返す。
  例： word ”ab cd ef” .....3 を返す
- 指定位置の文字を削除 (複数指定可) -v オプション
  例： word -v ”ab cd ef gh” 2,3..... ab gh を返す
- 右から位置指定する (一個しか指定出来ない)
  例： word -r ”ab cd ef gh” 2..... ef を返す
- デフォルトの区切りはスペース -F オプションで指定可能
  例： word -F: ”ab:cd:ef:gh” 2..... cd を返す

オプション一覧

```
*       :-F   区切り文字を指定
*       :-z   word 数が 0 以上のときその行を出力
*       :-v   指定位置の文字を削除 (複数指定可)
*       :-r   右から数える（一個しか指定出来ない)
*       :-n   以後オプション設定禁止
*       :-    標準入力を対象とする
*       :-d,  デバックモード
```

2.2　「word_add」語の追加、削除

- 文字列 1 に 文字列 1 に無い 文字列 2 を追加する
  例： word_add ”ab cd ef” ”aa cd ee” .....ab cd ef aa ee を返す
- 文字列 1 から 文字列 2 を削除する -r オプション
  例： word_add -r ”ab cd ef” ”aa cd ee” .....ab ef を返す
  以前の strcut 廃止はされ word_add -r と strswap に統合された。
  word_add -r は語 (区切り記号までが同じ) でないと削除されないが
  strswap は その文字列があれば削除することができる。

- デフォルトの区切りはスペース -F オプションで指定可能
  例： word_add -F: ”ab:cd:ef:gh” ”ij:ab”..... ab:cd:ef:gh:ij を返す

```
**********word_add**********Ver0401******************
#- 機　能　　: 文字列 に　文字列 を追加する
#-              ただし対象文字列中に追加文字列がある場合は追加しない。
#-利用法　　: 対象文字列　追加文字列　[オプション]
#-オプション: -F:　　区切り記号 (デフォルト スペース)
#             : -r　　対象文字列中 から　文字列を削除
#使用例　　: word_add "aa bb cc"  "11 222 aa" --->aa bb cc 11 222
************************************************************
```

使 用 例　:重複のないパスの追加

```
  PATH=‘/uap/BC/word\_add  -F: "$PATH" "/uhome/bin:./"‘ ; export PATH
```

使 用 例　:重複のないファイルの表示

```
  sh                                #-sh または bash でないと大きな変数が使えない
  L1=‘ls -1 /uap/BC|/uap/BC/vecread - -F’ ’‘ #- vecread で LF 等を取る
  L2=‘ls -1 ~/bin |/uap/BC/vecread - -F’ ’‘
  /uap/BC/word_add -r "$L1" "$L2"            #- $L1 にしかないものを求める
  Func Int Knjcount Let Print Readme a.out add1
```

2.3 「mid$」文字の切り出し

- 位置 長さ指定による文字の切り出し
  形 式:mid$ 文字列 位置 長さ
  例:mid$ ’test Text’ 6 2 ....Te を返す
  例:mid$ ’test Text’ 6 .....Text を返す 長さ指定がなければ全体

- 文字指定による文字の切り出し
  例:mid$ ’test (10 例:mid$ ’test (10

- 位置 長さ指定による切り出し文字のテスト
  例:mid$ ’test Text’ 6 2 Te ..... ステータス 0 を返す
  例:mid$ ’test Text’ 6 2 aa ..... ステータス 0 以外を返す

- 切り出し文字の置換 (切取)
  例:mid$ -c ’test Text’ 6 2 aa .....test aaxt を返す
  例:mid$ -c ’test (10 例:mid$ -c ’test Text’ 6 2 .....test xt を返す

```
*形　式:mid$(010317) 文字列 位置 長さ
*機　能:引数 1 の文字列の位置から長さだけ切り出す。
*使用例:mid\$ ’test Text’ 6     ....>Test
*      :mid\$ ’test Text’ 6  2  ....>Te
*      :mid\$ ’test Text’ 6  2 debag   切り出し文字のテスト..>??
```

```
*      :mid\$ 'test Text' 6  2  Te      切り出し文字のテスト..>0
*      :mid\$  -  6  2 .......>標準入力を処理する
*      :mid\$  -  6  2 -n    ..>行変えを出力しない。
* 位置指定が数字で始まらない文字列の場合はその文字列の後ろから
* 長さ指定が数字で始まらない文字列の場合はその文字列の前
*Option:-c      置換 (切取) モード 例: mid$  'a=32#'  'a=' # 55 -c ...> a=55#
*      :        置換文字列がないときは切取される
*      :-o      以後オプション設定禁止 (-が付いた文字列操作)
*      :-h      この案内            *      :-d      デバックモード
```

2.4 「right$」文字の切り出し

- 文字列の後ろから指定した長さを切り出す。
  例:right$'testText'4.....Text を返す
  例 : *right* 'test Text' 4 2 ..... Te を返す

- 文字列の後ろから指定した長さを切り出す。
  例:right$'testText'4.....Text を返す
  例 : *right* 'test Text' 4 2 ..... Te を返す

- 文字列の後ろから指定した長さを切りとる。 -r オプション
  例:right$'testText'4 − r.....test を返す
  例 : *right* 'test Text' 4 2 ..... Te を返す

- 文字列の後ろから最初に現れた文字以降を切り出す。
  例:right$'testText'T.....ext を返す
  例 : *right* 'test@nirs.go.jp' @ ..... nirs.go.jp を返す

```
* 形 式:right$(980709) 文字列 長さ
* 機 能:引数 1 の文字列の後ろから引数 2 長さだけ切り出す。
*使用例:right$ 'test Text' 2   .....> xt
*      :right$ 'test Text' 4  2  ...> Te
*      :right$ 'test Text' 6  2 -d デバックモード
*      :right$  - 6          .......>標準入力を処理する
*      :right$  - 6  -n        LF を出力しない  ...
*      :right$  - 6  -r        後の 6 文字を切り取る
*      :right$  as@ab@ccc  @          右の@以下出力  ...
```

2.5 「instr」文字の試験

- 文字列から文字列を検索しその位置を出力。
  例:instr '0c3132' 31 .....3 を返す

例:instr 'abcdefg' xy .....0 を返す
例:instr 'abcdefg' ab .....1 を返す
例:instr 'abcdefg' AB -i .....1 を返す
∵ -i により大文字小文字を区別しない
例:instr 'abcdefg' ab 2 .....0 を返す
∵位置指定が 2 となっているのでこれ以後には ab がない。

- 文字列から文字列を検索しその文字以降出力。
  例:instr 'abcdefg' bc -p .....bcdefg を返す
  例:instr 'abcdefg' bc -P .....defg を返す

- 文字列から文字列を検索しその文字以前を出力。
  例:instr 'abcdefg' bc -f .....abc を返す
  例:instr 'abcdefg' bc -F .....a を返す

- 文字列から文字列を検索しあれば全体を出力。
  例:instr 'abcdefg' bc -L.....abcdefg を返す
  例:instr 'abcdefg' bc -V ..... 何も返さない
  ∵ bc が見つかったので
  例:instr 'abcdefg' zz -V .....abcdefg を返す

- 文字列から文字列を検索しステータスを返す。
  例:instr 'abcdefg' bc -s..... ステータス 0 を返す。
  例:instr 'abcdefg' zz -s..... ステータス 0 以外を返す。

```
* 形 式:instr(000218) 文字列　検索文字列　位置 option
* 機 能:引数 1 の文字列から文字列を検索しその位置を出力。
*       :　見つからない場合は 0 を返す。
*使用例:instr '0c3132'　31
*      :instr '4142434441'　41 5
*      :instr '4142434445'　45 2 -d
*    option :-d,　デバックモード
*           :-p, 見つかった文字から出力
*           :-P　見つかった文字の直後から出力
*           :-f　見つかった文字まで出力
*           :-F　見つかった文字まで出力
*           :-L　文字列全体を出力
*           :-V　見つから無かったら文字列全体を出力
*           :-c　文字列の数を出力
*           :-i　大文字小文字を区別しない
*           :-s　ステータスを返す。
```

```
*       :-n  以後オプション設定禁止
*            :-   標準入力を対象とする
```

2.6 「strlen」 文字列の長さ

- 文字列の長さを返す。
  例:strlen 'test Text' .....9 を返す。

- 文字列の長さをテストしステータス返す。例:strlen 'test Text' -g5 ..... ステータス 0 を返す。
  ∵ 長さ 9 が 5 を越えているため
  例:strlen 'test Text' -g10 ..... ステータス 0 以外を返す。
  ∵ 長さ 9 が 10 を越えていないため

- 文字列の長さをテストし全体を返す。例:strlen 'test Text' -G5 .....test Text を返す。
  ∵ 長さ 9 が 5 を越えているため
  例:strlen 'test Text' -G10 ..... なにも返さない。
  ∵ 長さ 9 が 5 を越えていないため

```
**************strlen(021119)*****************
* 形 式:strlen  文字列
* 機 能:引数 1 の文字列の長さを返す。
*使用例:strlen 'test Text'  -->9
*       :strlen  -  標準入力を対象
*選択枝:-g 数字       数字以上の長さなら真 (0)
*       :-G 数字       数字以上の長さなら出力 - 指定はいらない
**************************************************
```

2.7 「strswap」 文字列の置き換え

- 文字列の中の第二引数の文字列を第 3 引数の文字列に変換
  例:strswap 'test Text' test TEST .....TEST Text を返す。例:strswap 'test Text' test ... . Text を返す。∵ 変換文字列がないので削除される。

- -f オプションで 変換ルールファイルを指定できる
  例:変換ルールファイル /uap/BC/data/Manth.wap

```
Jan
01
Feb
02
Mar
03
```

```
Apr
04
May
05
Jun
06
Jul
07
Aug
08
Sep
09
Oct
10
Nov
11
Dec
12
```

上記変換ファイルを用いると jan が 01 に Feb が 02 に変換される。

- 変換ファイルの確認
  例:strswap -t /uap/BC/data/Manth.wap

```
-------------変換対称文字列数--------変換元文字→置換文字-------------
Jan → 01   Feb → 02   Mar → 03   Apr → 04   May → 05   Jun → 06   Jul →
07   Aug → 08
Sep → 09   Oct → 10   Nov → 11   Dec → 12
```

『ベーシック風コマンド』 strswap(010802.2k) 文字列 文字列　文字列
形　式:strswap 文字列 文字列 文字列
機　能:引数 1 の文字列の中の第二引数の文字列を第 3 引数の文字列に変換
使用例:strswap 'test Text' test TEST
:strswap abcd ab .....ab をとる
:strswap - } : 標準入力を変換
:strswap -i dir.unx 'test:Text' 'test TEST'
:strswap -i file source distination -a 行の入れ換え
:strswap - a b ..... 標準入力を変換
:strswap -f 変換ルールファイル ..... 標準入力を変換
:strswap -t 変換ルールファイル ..... 変換ルールファイルの表示
オプション　-d デバックモード
-n オプション禁止

2.8 「string$」 返し数の文字列を出力

- 指定返し数の文字列を出力。
  形 式:string繰り返し数文字列
  例 : *string* 3 'abcd' .... abcdabcdabcd を返す。

```
* 形 式:string$ 繰り返し数　文字列
* 機 能:引数 1 繰り返し数の文字列を出力。
*使用例:string$  10 '-test-'
*       :string$  50 -
```

3 文字、文字列の変換

3.1 「toSmall」 小文字変換

- 文字列を小文字変換
  例:toSmall '0A31B3cd' .....0a31b3cd を返す。

- 大文字変換
  例:toSmall -r 'abcdefg' .....ABCDEFG を返す。

```
*   toSmall ----- ver 980709->030121-------*
* 形 式 :toSmall 文字列
* 機 能 :引数 1 の文字列を小文字変換
*         引数 1 が - のときは標準入力
*        :-r 小文字 大文字変換
* 使用例:toSmall '0A31B3cd'  ......0a31b3cd
*       :toSmall  -
```

3.2 「asc$」文字コードを出力。

- 文字コードを出力
  例:asc$ 'test Text' .....116 を返す

- 指定位置の文字コードを出力。
  例:asc$ 'test Text' 2 .....101 を返す
  ∵ e の文字列の位置のコードは十進で 101 である

連携

```
asc\$ 'test Text'  .......116
∵ t の文字列の位置のコードは十進で 116
hex\$ 116         .........74
∵ 十進の 116 は 16 進で 74
chr\$ 74
```

∵ 16 進の 74 文字は t
* 形 式:asc$ 文字列 位置
* 機 能:引数 1 の文字列の位置のコードを出力。
*使用例:asc$ 'test Text'
*　　:asc$ 'test Text' 2
*　　:asc$ 'test Text' 6　[-x,-j,-d]
*　　:asc$　表示 -j
*　-x　ヘキサ表示
*　-j　JIS コード表示

3.3 「bin$ 」十進数を 2 進に変換

- 十進数を 2 進に変換
  例 :bin$ 1024 .....10000000000 を返す

- 連続十進数を 2 進に変換
  例 :bin$ 255.255.252.0 .....11111111 11111111 11111100 を返す
  例 :bin$ "16 8" .....10000 1000 を返す

- 十進数を 2 進パターン変換
  例 :bin$ -1# "16 7" .....#0000 ### を返す
  ∵ 1 を# に変換して出力
  例 :bin$ -0- "16 7" .....1—- 111 を返す
  ∵ 0 を- に変換して出力

```
*************bin$***********Ver011005******
*　機　能　: 十進数を 2 進に変換
*オプション: -F.　連続変換時の区切りを . に デフォルトはスペース
*　　:-1#　1 を#にする
*　　:-0-　0 を-にする
*　　: -　標準入力を対象とする
* 使 用 例 :bin$ 1024　bin$ 255.255.252.0
*********************************************
```

3.4 「chr$」 16 進数に対応する文字列出力

- 16 進文字列に対応する文字を返す。
  例:chr41.....*A* を返す。
  例 : *chr* 414243 .....ABC を返す。

```
************chr$********980421*****************
```

```
* 形 式:chr$ 16 進文字列
*********************************************
```

3.5 「hex$ 」十進数を 16 進数で出力

- 十進数を 16 進に変換
  例:hex$ '133' ...85
  例:hex$ '255' ...ff
  例:hex$ '512' ...200
  例:hex$ '133.63.32.1' ..85 3f 20 1

- -F オプション指定で出力セパレータを変え、2HEX 出力する
  例:hex$ -F: '133.63.32.1' ..85:3f:20:01

```
*********************************************************
* 形 式:hex$  十進数文字列
* 機 能:引数 1 の文字列の値を HEX で出力。
*使用例:hex$ '133'            ...85
*       :hex$ '133.63.32.1'    ..85 3f 20 1
*       :-F:  出力セパレータを:にする ... この場合は 2HEX 出力
*       :hex$  -              標準入力を変換
*********************************************************
```

3.6 「decimal」16 進数を十進数に変換

- 16 進数を十進数に変換
  例:decimal FF .....255 を返す例:decimal ff:ff:fc:0.....255 255 252 0 を返す例:decimal -F. ff:ff:fc:0.....255.255.252.0 を返す

```
************decimal***********Ver010126******
*  機  能  : 16 進数を十進数に変換
*オプション: -F.    連続変換時の区切りを . に デフォルトはスペース
*           : -      標準入力を対象とする
* 使 用 例 :decimal E2  decimal 08:00:20:a4:f6:e2
*********************************************
```

3.7 「chrcut」文字の削除

```
『ベーシック風コマンド』 chrcut(980709) 文字列 文字列
 形  式:chrcut 文字列 文字列
 機  能:引数 1 の文字列の中の第二引数の文字列をとる
 使用例:chrcut 'test Text'  tes ....t と e と s をとる
       :chrcut  abcd ab  .....a と b をとる
```

```
:chrcut  -     : 標準入力を対象
:chrcut  - -a  : 数字とスペース以外をカットする。
:chrcut  - -n  : 数字をカットする。
:chrcut  - -c  : コントロールコードをカットする。
:chrcut  - -M  : ラインフィードをカットする。
```

4 日付の処理

4.1 「no_of_days」西暦 0 年からの日数または 2 個の日数差を求める。

- 西暦 0 年からの日数を求める。
  例 :no_of_days 2005-01-01 .....732312 を返す。
  例 :no_of_days 1998 .....729754 を返す。
  ∵：年のみのときは 1 月 0 日として計算する
  例 :no_of_days today ..... 今日までの日数を返す.

- 2 個の日数差を求める。
  例:no_of_days 2000-01-01 2001-01-01 ..... 366 を返す。
  例:no_of_days from 2000-01-01 to 2000-01-03 ..... 2 を返す
  例:no_of_days 2000-01-01 today .....1847 を返した。

- 指定月の日数を出力
  例:no_of_days -s 2000-02 ..... 29 日を返す

- 曜日を計算する。例:no_of_days -w 2000-01-01 .....6 を返す
  例:no_of_days -W 2000-01-01 ..... 土 を返す

- 月数を計算する。no_of_days -M 2000-01-01 2001-01-01 .....12 を返す
  no_of_days -M 2000-01-01 2005-01-01 .....60 を返す
  no_of_days -M 2000-01-01 today .....60 を返した

```
*************no_of_days***********Ver021003+******
*  機  能  :  西暦 0 年からの日数または 2 個の日数差を求める。
* 利用法   : no_of_days  年-月-日 or today or 月-日-年
*          : no_of_days  (from) 年-月-日 (to) (年-月-日)
*          : デフォルトは 年-月-日 12 以上は自動判定 (区切りは-/ :;,)
*オプション: -s 指定月の日数を出力
*          : -w 曜日を計算する。
*          : -W [日月火水木金土] を出力
*          : -M 月単位で出力
* 使 用 例   :no_of_days 1988  or 2001-02-28 or 12-07-1995
```

```
*****************************************
```

4.2 「nextday」次の日または、指定された増減日数の期日を求める

- 次の日を求める例:nextday 2000-02-28 .....2000-2-29 を返す.
  例:nextday 2000-02-29 .....2000-3-1 を返す

- 指定された増減日数の期日を求める.
  例: nextday today -7 .....1-14-2005 を返した
  例: nextday 2005-01-05 7.....2005-1-12 を返す

```
************nextday**********Ver021219******
*  機  能  :  次の日または、指定された増減日数の期日を求める
* 利用法   : nextday  月-日-年
*          : nextday  月-日-年 増減日 (デフォルト 1 日)
*          : デフォルトは 年-月-日 12 以上は自動判定 (区切りは-/ :;,)
*オプション: -i  疑似日数 年*10000+月*100+日 出力   -d デバックモード
*          : -m 数字  数字の月を増減し期日を求める
* 使 用 例 :nextday  2001-02-28  2003/11/26 or 12-07-1995
*****************************************
```

4.3 「s2nengo」西暦を和暦に変換

```
************s2nengo**********Ver000914,011022,011220******
*  機  能  :  西暦を年号に変換する。
* 利用法   : s2nengo 西暦  or 月-日-年 (区切りは-/ :;,)
*オプション: -s(略語) -t(変換表)
*          : -w 曜日を計算する。 年は省略
*          : -W 日曜日は <font color=red>を付ける
*          : -n 終了時に次の日を求め出力
* 使 用 例   :s2nengo 1988  or  12-07-1995
*****************************************
例:s2nengo 2005-01-01  ..... 平成 17 年 1 月 1 日 を返す。
   s2nengo 01-12-2006  ..... 平成 18 年 1 月 12 日を返す。
   s2nengo -w 2005-01-10  ......1 月 10 日【月】 を返す。
   s2nengo -W 2005-01-09  ......<font color=red> 1 月 9 日【日】 .. を返す。
   s2nengo -n 2005-01-9   ......
平成 17 年 1 月 9 日
2005-1-10  を返す
```

4.4 「nengo2s」和暦を西暦に変換

```
************nengo2s**********Ver000914,011022,011220******
*  機  能  :  年号を西暦をに変換する。
* 利用法   : nengo2s 年号 S20 H13 または 昭和 20 平成 13
*オプション: -l 年号、西暦併記        昭和 19 年   1944
*           : -a 経過年も出力 1944 昭和 19 年   59
*           : -t -a 形式で今年まで連続 出力
* 使 用 例   :nengo2s H8 or   平成 8
*****************************************
例: nengo2s S19   ..............  1944 を返す
    nengo2s 平成 17 年 .............2005 を返す
    nengo2s 平成 18 年 1 月 12 日 .....2005-1-12 を返す
    nengo2s -l H17    ............. 平成 17 年   2005 を返す
    nengo2s -a H1     .............1989 平成  1 年   16 を返す
```

4.5 「calconv」年月日の変換 04Feb18 040218 20040218 → 2004-02-18

```
************calconv**********Ver990621B*********************
*  機  能  :  年月日の変換 04Feb18 040218 20040218 -->2004-02-18
*           :  5 月 22 日 1997 年-->1997-5-22  5 月 22 日 18:00 -->2004-5-22
*           :  デフォルト自動判定
*           :  ls -l のフォマットを 標準変換
*           :-   標準入力を変換
*           :-s  2004-01-26 のフォマットを 04Jan26 に
* 利用法   :calconv Jan  ...>数字に変換
*           :calconv  2   ...> Feb に変換
************************************************************
```

4.6 「timeconv」秒に変換 03:18:20 → 03*3600+18*60+20

```
************timecnv**********Ver0408B*********************
*  機  能  :  時間の変換  03:18:20 --->(03*60)+18*60+20
*           :  5 時 30 分 20 秒 ----->19820
*           :  デフォルト自動判定
*           :-   標準入力を変換
*                第一カラムが 03:18:20 形式ならその部分が秒に変換される。
* 利用法   :timecnv  12:30 ...>秒に変換
*           :timecnv  230   ...> 3:50 に変換
************************************************************
```

4.7 「ltime」1970 1 1 の秒数 を得る

- 1970 年 1 月 1 日から現在までの秒数を得る。
  例:ltime .....1106302735 を返した。

例:ltime 2000-01-01.....946652400 を返す
例:ltime2date 946652400 ..... 2000/0/1(6) 0:0:0 を返す

```
************ltime**********Ver'000106******
*  機  能  :  1970 1 1 の秒数 を得る( unix 時間は 1970 1 1 から)
*          :  引数の指定がある場合はその日までの秒数
*  形  式  :  ltime [年[-月[-日-[-時-分]]]] 省略 1月1日0時0分
*          :  -d  デバックモード
* 関連     :  /uap/BC/ltime2date に渡すことにより日付に変更される
*****************************************
```

4.8 「ltime2date」1900-01-01 00:00:00 からの秒数 (UNIX time) を日付に

- 1900-01-01 00:00:00 からの秒数 (UNIX time) を日付に変換
  例 : ltime2date 1034811713 .....2002/9/17(4) 8:41:53 を得る
  例 : ltime2date 1034811713 -s ..... 2/9/17 8:41:53 を得る
  例 : ltime2date 1034811713 -l .....2002/9/17[木]8:41 を得る

```
************ltime2date**********Ver'021017************************
*  機  能  :  1900-01-01 00:00:00 からの秒数(UNIX time)を日付に
*オプション:  -s  年号を2桁表示
*          :  -l
*          :  -d  デバックモード
* 使 用 例 :  ltime2date 1034811713
*********************************************************************
```

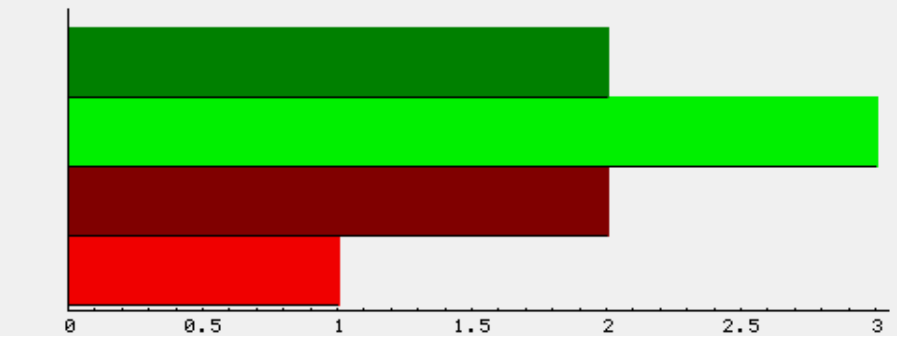
5 グラフ処理

いずれの処理も、グラフツール X ピックスマップインタープリタ (XPMi) を解してグラフを作っています。****4xpmi はいずれも XPMi が解釈できる命令語を作り出すものです。どれもスケールを書くため、XPMi を呼び出すとき、 -i1 オプションが必要です。

5.1 「bar4xpmi」棒グラフデータの作成

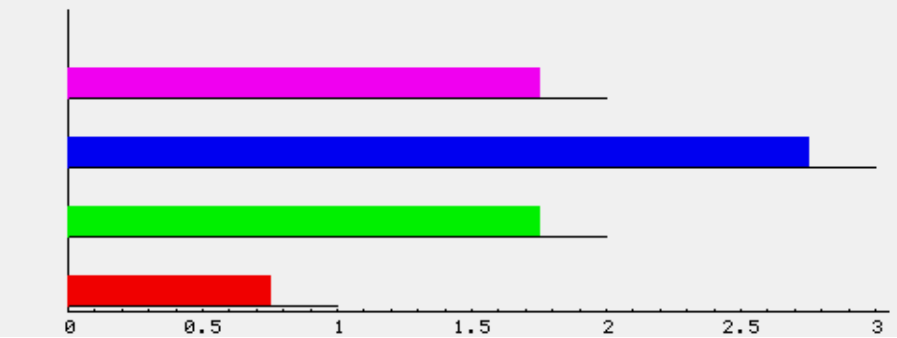
```
************ber4xpmi**********Ver021205******
*  機  能  :XPM インタープリター用棒グラフデータの作成
* 利用法   : [項] 値  のデータを標準入力からとる
*オプション: -cRGB     ..... 棒の色の順  rgbylRGBY
*          : -p0      ..... 横の棒グラフ(データの数が10個以下は自動で)
*          : -p1      ..... 縦の棒グラフ(データの数が10個以上は自動で)
*          : -k       ..... 項目名を書く
```

```
*          : -t       ... 右上にタイトルを書く
*          : -d       ..... デバック
* 使 用 例 :ber4xpmi 480  100    ....480*100 の絵を作る。
*          :ber4xpmi 480  500 -p1 | XPMi 480 500 -i1 | xpmtoppm | ppmtogif
cat bar
a 1
b 2
c 3
d 2

cat bar |bar4xpmi | XPMi -i1 | xv -
```



5.2 「ybar4xpmi」棒グラフデータの作成

```
*************yber4xpmi***********Ver011012******
*  機  能  :XPM インタープリター用グラフデータの作成
* 利用法   : x,y1,y2 データを標準入力からとる
*オプション: -w8      ......8 ドットの棒
*          : -cRGB     ..... 棒 (点) の色の順  rgbylRGBY
*          : -p1      ..... ポイントプロット -p0.... 棒 (デフォルト)
*          : -p2      ..... ポイントプロット+線
*          : -s2      ..... 複数のデータの場合のシフト (1)
*          : -d       ..... デバック
* 使 用 例 :yber4xpmi 700  500 -w8  ....700*500 の絵を作る。
*          :yber4xpmi 700  500 -w8 | XPMi 700 500 -i1 | xpmtoppm | ppmtogif
*****************************************
cat bar |yber4xpmi | XPMi -i1 |xv -
```



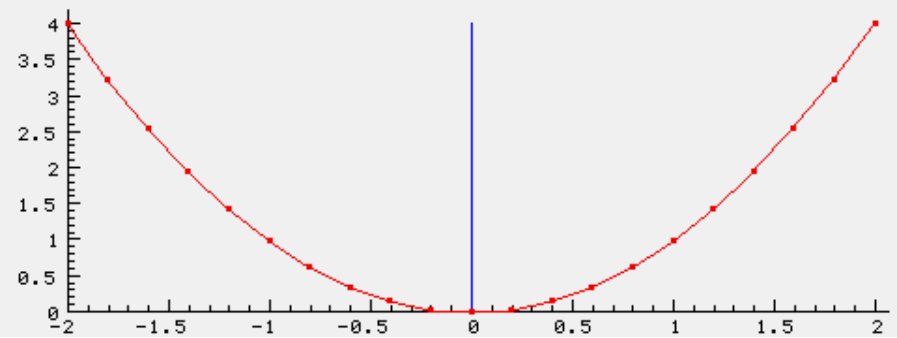
5.3 「zone4xpmi」帯グラフデータの作成

```
*************zone4xpmi***********Ver011012******
*  機　能　:XPM インタープリター用帯グラフデータの作成
* 利用法　 : (項) 値　 データを標準入力からとる
*オプション:
*          : -cRGB　　..... 帯の色の順　rgbylRGBY
*          : -d　　..... デバック
* 使 用 例 :zone4xpmi 480　50　....480*50 の帯グラフデータを作る。
*          :zone4xpmi 480　50　| XPMi 480 50 | xpmtoppm | ppmtogif
*****************************************
cat bar |zone4xpmi |XPMi -i1 |xv -
```

5.4 「plot4xpmi」XY グラフデータの作成

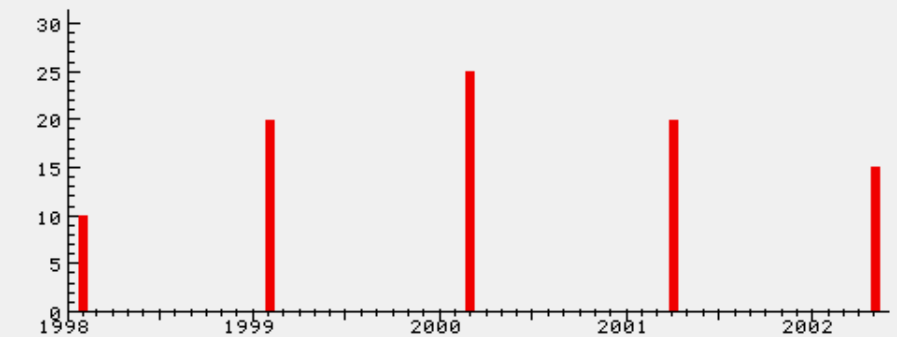
```
*************plot4xpmi***********Ver011012******
*  機　能　:XPM インタープリター用グラフデータの作成
* 利用法　 : x,y1,y2 (最大 20) データを標準入力からとる
*オプション: -w8　　.....8 ドットの棒
*          : -cRGB　　..... 棒 (点) の色の順　rgbylRGBY
*          : -p1　　..... ポイントプロット -p0.... 棒 (デフォルト)
*          : -p2,-p20 ..... ポイントプロット+線 20 戻りは PENUP
*          : -s2　　..... 複数のデータの場合のシフト (1)
*          : -d　　..... デバック
* 使 用 例 :plot4xpmi 700　500 -w8　....700*500 の絵を作る。
```

```
*          :plot4xpmi 700  500 -w8 | XPMi 700 500 -i1 | xpmtoppm | ppmtogif
*****************************************
 Func X^2 -2,2 | plot4xpmi -p2 |XPMi -i1 |xv -
```



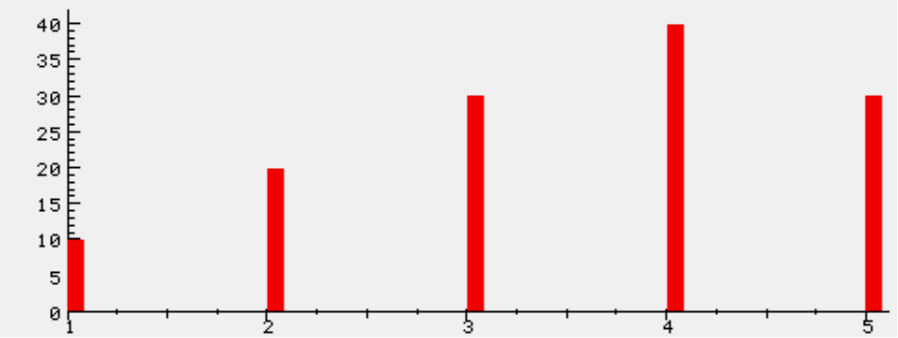
5.5 「date4xpmi」日付グラフデータの作成

```
*************date4xpmi***********Ver011113->021226******
*  機  能  :XPM インタープリター用日付グラフデータの作成
* 利用法   : date,y1,y2   データを標準入力からとる（後から順に書く）
*          : 日付の区切りは -/ データの区切りは ,;:
*オプション: -w8      ......8 ドットの棒
*          : -cRGB     ..... 棒（点）の色の順  rgbylRGBY
*          : -p1      ..... ポイントプロット -p0.... 棒（デフォルト）
*          : -p2      ..... ポイントプロット+線
*          : -s2      ..... 複数のデータの場合のシフト（1）
*          : -Y3      ......Y 軸をログスケールの 3 桁に -Y1 はリニア
*          : -t       ..... 右上にタイトルを書く
*          : -d       ..... デバック
* 使 用 例 :date4xpmi 700  500 -w8  ....700*500 の絵を作る。
*          :date4xpmi 700  500 -w8 | XPMi 700 500 -i1 | xpmtoppm | ppmtogif
*****************************************
cat date2
1998-02-01 10
1999-02-01 20
2000-03-01 25
2001-04-01 20
2002-05-01 15
cat date2 |date4xpmi -w5| XPMi -i1 |xv -
```



5.6 「day4xpmi」日:時 グラフデータの作成

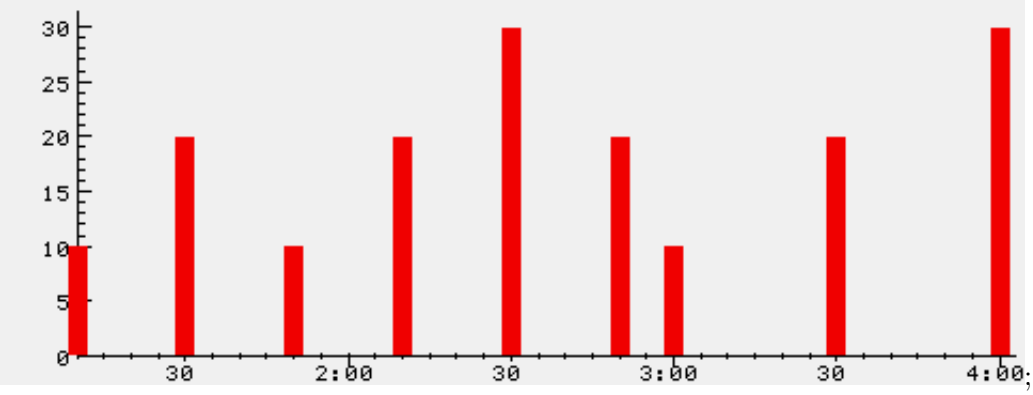
```
*************day4xpmi***********Ver040713a*********************************
*  機  能  :XPM インタープリター用 日:時 グラフデータの作成
*データ形式: day:hour,y1,y2    データを標準入力からとる（後から順に書く）
*           : 日時の区切りは /:  28:15 28-15  ←28 日 15 時
*           :           時間が戻った場合は最大日*24 時間たされます。
*           : データの区切りは ,;| （データ欠如は 0.0）
*オプション: -w8      ......8 ドットの棒
*           : -cRGB     ..... 棒（点）の色の順  rgbylRGBY
*           : -p1      ..... ポイントプロット -p0.... 棒（デフォルト）
*           : -p2      ..... ポイントプロット+線
*           : -s2      ..... 複数のデータの場合のシフト（1）
*           : -Y3      ......Y 軸をログスケールの 3 桁に -Y1 はリニア
*           : -F:      ..... 日時出力の区切り
*           : -t       ..... 右上にタイトルを書く
*           : -r       ..... 時間が戻った場合の処理をしない（random-data）
*           : -d       ..... デバック
* 使 用 例 :day4xpmi 700  500 -w8  ....700*500 の絵を作る。
*           :day4xpmi 700  500 -w8 | XPMi 700 500 -i1 | xpmtoppm | ppmtogif
* 分岐点    : 2 日、2 週間 で スケールの書き方が変わる
*************************************************************************
日時は原則として未来に向かってソートされていることが必要です。
cat ps/day2
1:1 10
2:1 20
3:1 30
4:1 40
5:1 30

cat day2 | day4xpmi  -w8|XPMi -i1 |xv -
```



;

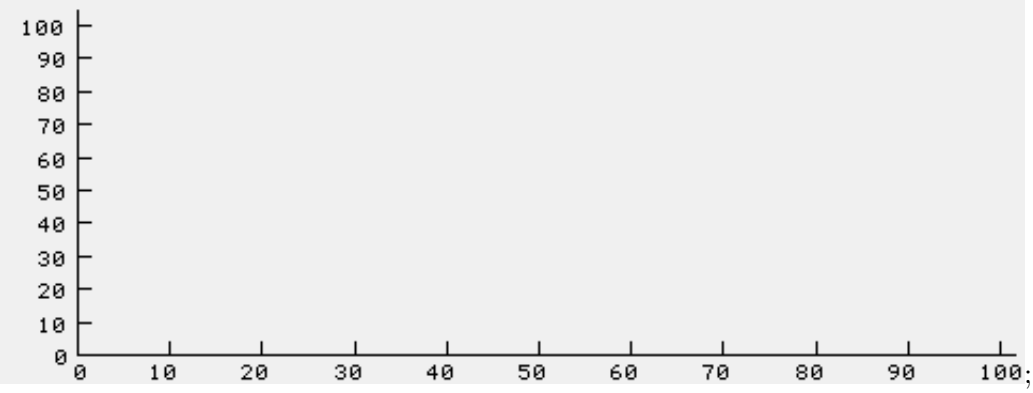
5.7 「time4xpmi」時間グラフデータの作成

```
*************time4xpmi***********Ver011113->021226A******
*  機　能　:XPM インタープリター用 時間グラフデータの作成
* 利用法　 : time,y1,y2　 データを標準入力からとる（後から順に書く）
*          : 時間の区切りは /:| データの区切りは ,;: （データ欠如は 0.0）
*オプション: -w8      ......8 ドットの棒
*          : -cRGB     ..... 棒（点）の色の順　rgbylRGBY
*          : -p1      ..... ポイントプロット -p0.... 棒（デフォルト）
*          : -p2      ..... ポイントプロット+線
*          : -s2      ..... 複数のデータの場合のシフト（1）
*          : -Y3      ......Y 軸をログスケールの 3 桁に -Y1 はリニア
*          : -t       ..... 右上にタイトルを書く
*          : -r       ......randam data
*          : -d       ..... デバック
* 使 用 例 :time4xpmi 700  500 -w8  ....700*500 の絵を作る。
*          :time4xpmi 700  500 -w8 | XPMi 700 500 -i1 | xpmtoppm | ppmtogif
*******************************************
cat time1
1:10 10
1:30 20
1:50 10
2:10 20
2:30 30
2:50 20
3:0  10
3:30 20
4:0  30
cat time1 |time4xpmi -w8 |XPMi -i1 |xv -
```



5.8 「scale4xpmi」スケールの作成

```
**********scale4xpmi**********Ver030602******
*  機  能  :XPM インタープリター用スケールの作成
* 利用法   : 座標 xmin ymin xmax ymax [xsize ysize ] をアギュメント指定
*オプション:
*          : -X3      ......X 軸をログスケールの 3 桁に xmax 優先
*          : -Y3      ......Y 軸をログスケールの 3 桁に ymax 優先
*          : -d       ...... デバック
* 使 用 例 :scale4xpmi 0 0 10 10 700     ....(0,0)-(10,10) 700*500 のスケールを作る。

*          :scale4xpmi 0 0 100 100 | XPMi -i1 |xv -
```



5.9 「XPMi」XPM インタープリター

XPMi は -h オプションで簡単な help が表示されます。

```
**********XPMi**********Ver011012**********
*  機  能  : XPM インタープリター (800x500) 標準 (480x160)
*          : XPM  X 長さ Y 長さ [bgcolor] オプション
*オプション: -b バックグランドカラーの指定.
*          : -f フォアグランドカラーの指定#
*          : -x[n]  原点 X
*          : -X
```

```
*          : -y  原点 Y
*          : -Y
*          : -i 初期モード 0 nomal 1:scale1(36,14)  2:scale2
*-------- インプリ命令------一文字命令-----------------------
*m move x y   ;p point x y      ; -,h hline y l ; |,v vline x l
*a x y ASC              #コメント行
*s x y 文字  ;S x y 文字        ;. 横鎖線  : 縦鎖線   & 連続相対直線
*/,l line x y  ;c color(.rRgGbBmMyYl#);  大文字は暗い色
* 使 用 例   :XPMi 400 300 g -i1  (原点=36,14)
*-----------------------------------------------------------------*
```

XPMi は標準入力からコマンドを
受け取ってピクセルを作り上げます。 おもなコマンドは

移動コマンド m x y

点描画	p x y
水平線	h y l or - y l
垂直線	v x l or | x l
水平鎖線	. y l
垂直鎖線	: x l
直線	l x y or / x y
連続相対直線	& x,y x,y x,y
色指定	c [.rRgGbBmy1#]

XPMi が扱う最大キャンバスは 800x500 です。デフォルトは 480x160 になっています。 カラーは
右の 13 色です。
XPMi を起動すると標準入力待ちになります。

m 30,30
は 30,30 にポインターを移動することを意味します。
& x1,y1 x2,y2 x3,y3 ..... 等のように連続相対線を描く起点を決めるのに用います。

- 30 50
は y 座標 (下からのドット数) が 30 で長さが 50 ドットの線を x の 0 座標から引きます。
同様に . 30 50 で点線 (1 ドットおき) を引きます。
| 50,100
は x 座標 (下からのドット数) が 50 で長さが 100 ドットの線を y の 0 座標から引きます。

: 50,100 で上記の点線です。

cg
はカラーを緑に変更する命令です。色は赤、緑、青をあらわす rgb とやや暗い RGB
y,m,l は黄, マゼンダ, 水色これより暗い Y,M があります。また .#はそれぞれ白黒です。

a 50,0 ASC 文字
S 50,0 文字

s 50,0 文字
は 絶対座標 50,0 に文字を書くものです S,s コマンドで漢字を書くことができるのですが
24 ドット漢字フォントしか持っていないため s コマンドで半分に間引きして 12 ドットで
表示
できるようにしてあります。
その他
終点を与えて 起点から斜線を書く / x,y
終点を与えて 起点から四角を書く b x,y
点をプロットする p x,y があります。

6 数値および表処理

6.1 「add1」引数 1 に 1(または引数 2) を加算して出力。

- 引数 1 に 1(または引数 2) を加算して出力。
  例:add1 1 .....2 を返す。
  例:add1 5 2 .....7 を返す。

- 0 から設定数未満 でを繰り返す。例:add1 5 -m6 ..... 0 を返す。
  例:add1 4 -m6 ..... 5 を返す。
  例:add1 5 2 -m6 .....1 を返す。

- 1 から設定数までを繰り返す。
  例:add1 5 -M6 .....6 を返す。
  例:add1 5 2 -M6 .....1 を返す。

- 00 のついた数字を返す
  例:add1 001 -l3 .....002 を返す。

```
************add1********0211************
* 形 式:add1 数字
* 機 能:引数 1 に 1(または引数 2) を加算して出力。
*使用例:add1 1     ...2  add1 5 2  -m6 ......
*オプション: -m 整数  mod の設定 整数以上の場合は 0 に
*           : -M 整数  mod の設定 整数以上の場合は 1 に
*           : -l 整数  001 の様に 長さを指定する (exs -l3) 12 桁以内
************************************************
```

6.2 「Let」引数を数学処理して出力。

```
*************Let,Int***********Ver2011<-981213******************
#- 形 式:Let,Int 引数
#- 機 能:引数を数学処理して出力。
#-使用例:Let  10+20
#-       :Let $A+32.3x2
```

```
#-Int     整数出力
#-LET ... デバックコマンド
#-関数 exp(),erf(),log(),LOG(),sin(),cos(),tan(),sqrt(),p10(),gamma(),strlen()
#-演算  *,/,+,-,<,>,==,=,@,!=,!,%,and,&,or,o,||
#-  英大文字変数は  /uap/BC/data/values に定数
```

6.3 「Int」引数を数学処理して整数出力。

Let と同一で名前が Int なら整数出力する。
例:Int ”5/(1+2)” .....1 を返す
例:Int ”sin(3.1417)”..... 0 を返す
例:Int 10/3 .....3 を返す

6.4 「mat」表 (数値, 文字) をディメンションのように扱う

- 文字マトリックスの切り出し

  例: M="1,jan 2,Feb 3,Mar" のとき
  ```
  mat "$M" 1 ..... 1,jan を返す
  mat "$M" 1.2 .....jan を返す
  ```

- マトリックスの最大値、最小値

  例: M="1,10 2,20 3,30" のとき
  ```
  mat "$M" max ..... 3,30 を返す
  mat "$M" min .....1,10 を返す
  mat "$M" max.1 .....3 を返す
  mat "$M" max.2 .....30 を返す
  mat "$M" 2.max .....20 を返す
  ```

- マトリックスの合計値、平均値

  例: M="1,10 2,20 3,30" のとき
  ```
  mat "$M" sum .....  6,60   列合計を返す
  mat "$M" sum.2  .....60  2 列目の合計を返す
  mat "$M" 0.sum .....
  11
  22
  33  行合計を返す
  mat "$M" mean .....2,20 列平均を返す
  mat "$M" 0.mean .....
  5.5
  11
  16.5 行平均を返す
  ```

- マトリックスの作成, 代入

例: M=‘mat -i 3,3‘ .....0,0,0 0,0,0 0,0,0 を返す
例: mat "$M" 1.1 123 .....123,0,0 0,0,0 0,0,0 を返す

```
*************mat***********Ver000602F(30000 文字)**********************
*  機  能  : 文字列をディメンションのように扱う (形式 , スペース区切り)
* 利用法   : mat マトリックス  位置 (縦. 横)
*          :     位置は 1(1 行を表示)  2.1  0.2(2 列を表示) 0 は全体
*          :     max,min  でその行または列の最大、最小を表示
*          :     sum,mean でその行または列の合計, 平均値を表示
*          :     位置指定が無い場合は最大デイションを表示
*          :mat  マトリックス 位置 (縦. 横) 値  値をマトリックスに代入
*オプション: -   標準入力 (標準形式 および 複数行対応)
*          : -t  転置=縦横 がえしのマトリックス -tn の場合は複数行表示
*          : -i[初期値 [0]]  縦, 横 の行列を出力
*          : -F: 内部セパレターの変更 (通常は,)
*          : -d デバックモード
* 使 用 例 :mat "abc,1 efg,2 hij,3"  2.1
************************************************************
休日を求める例題 : 2? は 2 周目の月曜日を表すこととする。
N=20$1
HOLIDY="01-01, 元旦 \
01-2?, 成人の日 \
02-11, 建国記念日 \
03-21, 春分の日 \
04-29, みどりの日 \
05-03, 憲法記念日 \
05-05, こどもの日 \
07-3?, うみの日 \
09-3?, 敬老の日 \
09-23, 秋分の日 \
10-2?, 体育の日 \
11-03, 文化の日 \
11-23, 勤労感謝の日 \
12-23, 天皇誕生日"
WD="日, 月, 火, 水, 木, 金, 土"
DAYS=‘mat "$HOLIDY" 0.1 | strswap - ’,’ ’ ’ ‘
i=1
for D in $DAYS
do
        F=‘right$ $D 1‘                          # -最後の文字--
        m=‘word -F- $D 1‘                        # -月----
        d=‘word -F- $D 2‘                        # -日 または週--
        [ "$F" = "?" ] && d=‘/data/sys/get/monday $N $m $d‘
        W=‘no_of_days -w $N-$m-$d‘               #- 曜日は
        W1=‘add1 $W‘
        echo "$N-$m-$d ‘mat "$HOLIDY" $i.2‘[‘mat "$WD" 1.$W1‘]"
        [ $W -eq 0 ] && echo "‘nextday $N-$m-$d‘ 振替休日 [月]"
        i=‘add1 $i‘
done
.............. 出力.........................
2006-01-01 元旦 [日]
2006-1-2 振替休日 [月]
```

```
2006-01-9 成人の日 [月]
2006-02-11 建国記念日 [土]
2006-03-21 春分の日 [火]
2006-04-29 みどりの日 [土]
2006-05-03 憲法記念日 [水]
2006-05-05 こどもの日 [金]
2006-07-17 うみの日 [月]
2006-09-18 敬老の日 [月]
2006-09-23 秋分の日 [土]
2006-10-9 体育の日 [月]
2006-11-03 文化の日 [金]
2006-11-23 勤労感謝の日 [木]
2006-12-23 天皇誕生日 [土]
```

6.5 「matinv」逆行列を求める。

```
例:matinv  "2,1 3,2"  ..... 2,-1 -3,2 を返す
例:matinv  "1,2 1,2" .....inf,-inf -inf,inf を返す
例:matinv  "1,2 3,4" -d
Input Matrix=1,2 3,4
        1.00        2.00
        3.00        4.00
Inverse Matrix=
       -2.00        1.00
        1.50       -0.50      を返す
```

連立方程式を解く

```
     5x+4y=22
     2x+3y=13 の連立方程式を解くには、

    | 5 4 |* |x y| = |22 13| と行列表現し
    | 2 3 |

 /uap/BC/matinv "5,4 2,3" を実行し、逆行列を得る。
 0.428571,-0.571429 -0.285714,0.714286 を得る。
/uap/BC/matvec "0.428571,-0.571429 -0.285714,0.714286" "22,13" -m
  により 2.00        3.00 を得る.

matvec "`matinv "5,4 2,3"`" "22,13" のように書いてもよい。
*************matinv***********Ver01407******
*  機  能  : 逆行列を求める。
* 利用法   : matinv  行列
* 使 用 例 : matinv  "2,1 3,2"
*オプション: -m   ..... 行列表示
*          : -d   ..... 冗長表示
*          : -0   最短長１行出力（デフォルト）
*          : -1   最短長行列出力
```

```
*           : -2    f12.2 行列出力
*****************************************
```

6.6 「matmat」行列 (64) x 行列を求める。

- 行列積を求める。
  例：matmat ”2,1 3,2” ”1,2 4,5” .....6,9 11,16 を返す

- 各行, 列演算する。
  例：matmat ”2,1 3,2” ”1,2 4,5” -s+ .....3,3 7,7 を返す
  例：matmat ”2,1 3,2” ”1,2 4,5” -sx .....2,2 12,10 を返す

```
************matmat**********Ver010409=>030922*****************
*  機  能   : 行列 (64)X 行列を求める。
* 利用法    : matmat  行列 ベクトル
* 使 用 例 : matmat  "2,1 3,2" "1,2 4,5"
*オプション: -m    ..... 行列表示
*           : -s(x,/,+,-) ... 各行, 列演算
*           : -d    ..... 冗長表示
*           : -o(0,1,2,3,4)    （デフォルト 0）
*           :    0   g,     1 行出力       -1   g,   n 行出力
*           :    1   6g     n 行出力        2   12g   n 行出力
*           :    3   0,f6.3 n 行出力        4   0,-,* n 行出力
*                5   12.2f n 行出力
************************************************************
```

6.7 「matvec」行列 (表) とベクトル (列) の演算

- 行列ベクトル積
  例: matvec ”2,1 3,2” ”1,2” ..... 4,7 を返す。

- 各行演算
  例: matvec ”2,1 3,2” ”1,2” -sx ..... 2,1 6,4 を返す
  ∵: 1 行目に 1 を掛け、2 行目に 2 を掛ける
  例: matvec ”2,1 3,2” ”1,2” -s+ ..... 3,2 5,4 を返す
  ∵: 1 行目に 1 を加え、2 行目に 2 加える

```
************matvec**********Ver010407->0409******
*  機  能   : 行列 X ベクトルを求める。(128,32)=30000 文字
* 利用法    : matvec  行列 ベクトル
* 使 用 例 : matvec  "2,1 3,2" "1,2"
*オプション: -m(-1,0,1,2)    ..... 数字行列表示の変更 (0)
*           : -s(x,/,+,-) ... 各行演算
*           : -a(t,b,l,r) ... 上下左右の行の追加
*           : -     行列の標準入力
```

```
*          : -d    ..... 冗長表示
*****************************************
利用例: 表に列や行を追加する。
>NAME="A,B,C"
>MAT="10,11,12 \
15,14,13"
>SUM=‘mat "$MAT" sum‘
>HYO=‘matvec -at "$MAT" "$NAME"‘
>HYO=‘matvec -ab "$HYO" "$SUM"‘
>HYO=‘matvec -al "$HYO" "氏名,1 回,2 回, 合計"‘
>matprint -S "$HYO"
氏名     A        B        C
1 回      10       11       12
2 回      15       14       13
合計     25       25       25    ..... を得る。
```

6.8 「matsca」ベクトル (or) マトリクスとスカラーの演算

- ベクトル (or) マトリクスとスカラーの演算
  例 :matsca ”1,2,3” x 3 .....3,6,9 を返す。
  例 :matsca ”1,2 3,4” + 3 .....4,5 6,7 を返す。

- 各要素の逆数を求める。
  例 :matsca ”1,2 3,4” -r .....1,0.5 0.333333,0.25 を返す。

```
************vecsca**********Ver030905******
*  機  能  : ベクトル (or) マトリクスとスカラーの演算
* 利用法   : matsca ベクトル 演算記号 (+-x/) スカラー (default=1)
*          : matsca スカラー 演算記号 (+-x/) ベクトル
*オプション: -r 逆数
*          : -F: 内部セパレターの変更 (通常は,)
* 使 用 例 :vecsca  "1,2,3" -r
*****************************************
```

6.9 「matfrac」行列の分割された一部を出力する。

長過ぎる行列の出力に使う

- 横に長過ぎる行列を分割する。
  matfrac ”2,1,0,-1 3,2,2,3” 2 1 .....2,1 3,2 を返す
  matfrac ”2,1,0,-1 3,2,2,3” 2 2 .....0,-1 2,3 を返す

```
************matfrac**********Ver030928**********************
*  機  能  : 行列の分割された一部を出力する。
* 利用法   : matfrac  行列 横 (列) 分割数 出力位置
```

```
* 使 用 例 : matfrac  "2,1,0,-1 3,2,2,3" 2 1
*オプション:
*          :-d   ..... デバックモード
*****************************************************************
```

6.10 「matread」mat が扱えるデータフォマットに変換

表のタイトル等を省いてマトリックスを出力する。

```
*************matread***********Ver030905B******
*  機  能  : mat が扱えるデータフォマットに変換
* 利用法   : matread ファイル名 オプション
*オプション: -lx   左カラムを x 個とる
*          : -tx   トップから x 行とる
*          : -c#   コメント行の指定
*          : -Cx    カラムの数指定
*          : -F: 行セパレターの変更 (通常はスペース)
*          : -d デバックモード
* 使 用 例 :matread  /uap/ides/data/ee/1-3.50
*****************************************
```

6.11 「matprint」行列 (64)or 連立方程式を出力する。

```
*************matprint***********Ver010409>0309c**********************
*  機  能  : 行列 (64)or 連立方程式を出力する。
* 利用法   : matprint  行列
* 使 用 例 : matprint  "2,1 3,2"
*          : matprint  "2,1 3,2" "5 6"...2x + 1y = 5
*                                        3x + 2y = 6
*オプション: -s,-S   文字列のまま出力 (, またはタブ区切り)
*          : -H      HTML 形式で出力  -Hn <table>を出力しない。
*          : -       標準入力
*          :-d   ..... デバックモード
*          : -o(0,1,2,3,4)   (デフォルト 2)
*          :    0   g,     1 行出力       1  g,    n 行出力
*          :    2   6g     n 行出力       3  12g   n 行出力
*          :    4   0,f6.3 n 行出力       5  0,-,* n 行出力
*               6  12.2f   n 行出力
*************************************************************
利用例: HTML の表を出録
>matprint "a,b,c 1,2,3" -H
<table>
<tr>
 <td>a</td>
```

```
 <td>b</td>
 <td>c</td>
</tr>
<tr>
 <td align=right>1</td>
 <td align=right>2</td>
 <td align=right>3</td>
</tr>
</table>   ..... を得る
```

6.12 「vecvec」ベクトルとベクトルの演算

- ベクトル積
  例 :vecvec ”1,2,3” . ”3,4,5” .....26 を返す
  ∵ Let 1*3+2*4+3*5 26
- 各行演算
  例 :vecvec ”1,2,3” x ”3,4,5” .....3,8,15 を返す
  例 :vecvec ”1,2,3” + ”3,4,5” .....4,6,8 を返す

```
**********vecvec**********Ver030905A******
*   機   能   :  ベクトルとベクトルの演算
*  利用法     :  vecvec  ベクトル  演算記号 (+-x/.)  ベクトル (default=1)
*               :  .  はベクトル積   他は各行演算
*               :  -r  逆数
*               :  -F:  内部セパレターの変更 (通常は,)
*  使 用 例  :vecvec   "1,2,3"  x  "3,4,5"
*****************************************
```

利用例 : 単価と数量から合計計算

```
     hinmei="りんご, ミカン, バナナ"
     tanka="50,20,100"
     suuryo="5,10,6"
     /uap/BC/vecvec $tanka . $suuryo
```

1050 ..... を得る。

```
    MAT="品名,$hinmei 単価,$tanka 数量,$suuryo 合計,‘/uap/BC/vecvec $tanka . $suuryo‘"
    /uap/BC/matprint -S "$MAT"
```

品名	りんご	ミカン	バナナ
単価	50	20	100
数量	5	10	6
合計	1050	..... を得る	

6.13 「vecsort」行列 (64) の並び替え (標準化)

```
**********vecnorm**********Ver030911A**********
```

```
*  機  能  : 行列 (64) の標準化 (引数２に従って並び変え.. 同じものは合計される)
* 利用法   : vecnorm  行列 (項目、値) ベクトル (値)
* 使 用 例 : vecnorm  "A,1 D,3 C,4" "A,B,C,D"
*          :      A,B,C,D に対応して  1,0,4,3 を得る
*          : -    行列の標準入力
*          : -m[1, ]   1 は強制的に数字モードにする。通常は最初の値で自動判定
*          : -d   ..... 冗長表示
*********************************************************
```

6.14 「vecsca」ベクトル (or) マトリクスとスカラーの演算

matsca に統合

6.15 「vecprintf」ベクトル (64) フォーマットを出力する。

```
************ vecprintf ***********[Ver031003]**********************
*  機  能 : ベクトル (64) フォーマットを出力する。
* 利用法  : vecprintf ベクトル  印刷フォーマットベクトル
*使 用 例 : vecprintf  "2,1,3" "%6s"  ..6 文字で繰り返し出力
*         : vecprintf  "2,1,3" "<tr><td>%s</td>,<td>%s</td>,<td>%s</td></tr>"
*           出力                <tr><td>2</td><td>1</td><td>3</td></tr>
*注意!!印刷フォーマットは直接 c の printf に渡るたのでエラーのときはコアダンプする
*オプション: -F   入力セパレターの変更 (通常は,)
*                 フォーマットベクトルのセパレターはコンマから変更出来ない。
*          : -f 実数出力 %g %G %E %e %f などでフォーマットする。
*          : -i 整数出力 0 でフォーマットする。
*          : -    ベクトルを標準入力          -d   ..... デバックモード
*************************************************************************
```

6.16 「vecprint」複数のベクトル (行) を縦ベクトル (列) で出力する。

```
例 :vecprint  "2,1,3,2"
2
1
3
2 を返す
例 :vecprint  "2,1,3,2" "5,6,7,9"
2       5
1       6
3       7
2       9 を返す

************vecprint***********Ver030409B******
*  機  能  : 複数のベクトルを縦ベクトルで出力する。縦ベクトル
```

```
* 利用法    : vecprint  行列 [行列] [行列]
* 使 用 例 : vecprint  "2,1,3,2"
*           : vecprint  "2,1,3,2" "5,6,7,9"
*オプション: -F    出力セパレターの変更 (通常はタブ)1 文字
*           : -     標準入力
*           : -n    番号付き出力
*           : -d    ..... デバックモード
*****************************************
```

6.17 「sum」標準入力からデータを受取り、合計を求める

```
************sum**********Ver020421******
*  機  能  :  標準入力からデータを受取り、合計を求める
* 利用法    :  /uap/BC/select 4,5,6,7,8 $D | sum -I
*           : sum /etc/passwdn
*オプション: -F セパレータを指定する ファイルは自動 デフォルトは |
*           : -n 最終値のみ印刷
*           : -i 最終値のみ整数で印刷
*           : -I +集計 最終値のみ整数で印刷
*           : -d -D デバックモード e=exit
* 使 用 例 :sum -F',' *******
*****************************************
```

6.18 「mean」平均値 と標準偏差を求める。

```
************mean**********Ver010502**********************
*  機  能  :  平均値 と標準偏差を求める。
* 利用法    :  mean "数字列"
* 使 用 例 :  mean "2 3 4"
*オプション: -     データを標準入力から
*オプション: -d    詳細表示   最大、最小も表示
通常は 平均値, 標準偏差   のみ出力
*********************************************************
```

6.19 「rand」時間または/tmp/rand から乱数をつくる。

```
* 形 式:rand(990309)
* 機 能:時間または/tmp/rand から乱数をつくる。
*  rand  50  .......50 までの整数を作る。
*  rand  50 1  ......50 までの整数に 1 を加える
* -h   このメッセジ
* -d   デバックモード
```

6.20 「hist」階級にわけヒストグラムをつくる

```
************ hist ***********Ver030708A*************************
* 機　能　 :階級にわけヒストグラムをつくる (-i 指定がなければ自動)
* 形　式　 :hist [ファイル]　 :一行 一数値のデータ
*オプション: -i 最小値, 最大値,[STEP]　 ....-i1,10
*　　　-d デバック
*　利用例　: hist ファイル | /uap/BC/plot4xpmi -w20 | XPMi -i1 |xv -
*********************************************************************
```

6.21 「val」文字列の数値を出力。

```
* 形 式:val 文字列
* 機 能:引数 1 の文字列の値を出力。
*使用例:val '12.5kg'          ...12.50
*      :val '350km''          ..350.00
*      :val  350km  int       ..350
*      :val  350.3240  %5.1f   ..350.3
*      :val  -           標準入力を変換
```

7　数学関数処理

7.1 「Func」関数値を出力。

```
*********  :Func   [011013] **********
* 形 式:Func  関数 値 [, 値][,STEP]
* 機 能:引数 1 の関数 値を出力。 STEP 10(デフォルト 20) ー 100
*使用例:Func sin 20   ... 度単位     sin(X)   0,3.14  ラジアン
*      :Func cos 0,90
*      :Func tan 0
*      :Func pai 1
*      :Func X*X+3*X+1 0,5  ......
```

7.2 「lsm1」一次の最小 2 乗法

```
*************lsm1***********Ver010404******
*　機　能　: 一次の最小 2 乗法
* 利用法　 :lsm1 2 次元マトリック
* 使 用 例 :lsm1 "2,2 3,3 4,4"
*オプション: -d 詳しく出力 通常は勾配と切片のみ出力
*　注意　　: 区切りは,:;、スペースが使えるが数があってないと思わぬ結果となる
*********************************************
```

7.3 「ldiffeq」差分行列 と初期ベクトルを与えて差分方程式を逐次計算

```
************ldiffeq**********Ver031003A-******
*  機  能  : 差分行列 と初期ベクトルを与えて差分方程式を逐次計算
* 利用法   : ldiffeq  差分行列 初期ベクトル 計算間隔,計算終了時
* 使 用 例 : ldiffeq  "2,1 3,2" "1,2"  0.1,10
*オプション: -on  ..... 出力数の変更 (デフォルト 50 個以内)
*          : -    差分行列の標準入力
*          : -d   ..... 冗長表示
*****************************************
```

7.4 「ldiffint」差分行列 と初期ベクトルを与えて差分方程式を逐次積分計算

```
************ldiffint**********Ver031003******
*  機  能  : 差分行列 と初期ベクトルを与えて差分方程式を逐次積分計算 (US)
* 利用法   : ldiffint  差分行列 初期ベクトル 計算間隔,計算終了時
* 使 用 例 : ldiffint  "2,1 3,2" "1,2"  0.1,10
*オプション: -on  ..... 出力数の変更 (デフォルト 最終値のみ)
*          : -    差分行列の標準入力
* 出力回数が指定されていなく計算回数が 10000 を越える場合は 1/10 以降は
*            dt が 10 倍される。
*          : -d   ..... 冗長表示
*****************************************
```

8 ファイル処理

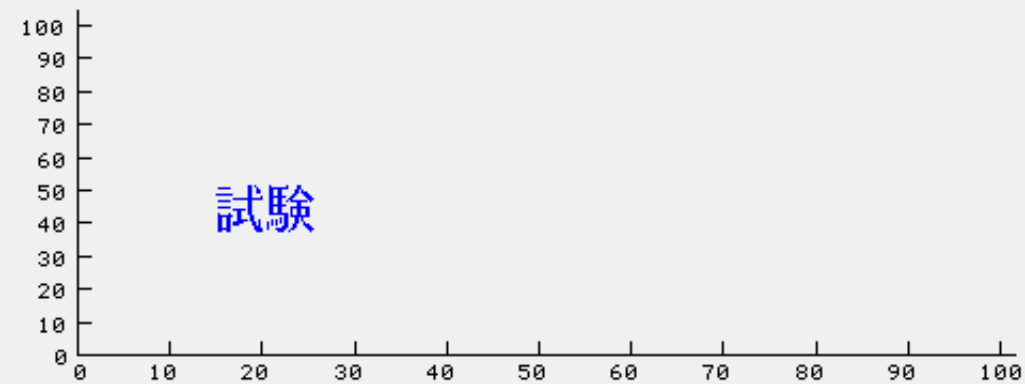
8.1 「addtop」標準入力をファイルの先頭に追加

```
cat file1
abcd
efgh
例 :echo "123" | addtop file1   ..........
123
abcd
efgh を返す
例 :echo "123" | addtop file1 -w ........file1 が追加更新される。
************addtop**********Ver000209B******************
*  機  能  :  標準入力をファイルの先頭に追加
*          : -a  標準入力の先頭にアギュメントを追加 (改行無し)
*          : -b  標準入力の先頭行にアギュメントを追加
* 利用法   : addtop  ファイル [オプション]
*          : addtop -a head,    ....head, が標準入力の先頭に追加
*オプション: -a   -h=help   -d=debag-mode
*          : -w 書き戻し 指定がない場合は標準出力
```

```
*           : -l   出力の長さ制限 (default=100 行)
************************************************************
```

8.2 「addtail」標準入力の後にアギュメントを追加

例 : echo "123" | addtail "abc" "def" .....
123
abc
def が返される
例 : scale4xpmi 0 0 100 10| addtail "S 100,70 試験" |XPMi -i1 | xv -




```
*************addtail***********Ver021031******
*  機   能   :  標準入力の後にアギュメントを追加 （全体の後のみ）
*           : -f|-w ファイル .. ファイルの後ろに標準入力を追加
*             -f 標準出力|-w ファイル出力
* 利用法    : addtail   追加文 1 追加文 2 追加文 3
*オプション: -1 一行ごとに行末にアギュメントを追加
*           : -l100   -f|-w モード時の長さ
*           : -h=help    -d=debag-mode   -l
*********************************************
```

8.3 「fline」『ファイル』の 『行番号』の前後『Dx』を表示

```
************************************************************
* 関数名:fline(991006)  ファイル名 行番号  [行間 Dx]
          『ファイル』の   『行番号』の前後『Dx』を表示
          『ファイル名』が 『-』 のときは標準入力
          『ファイル名』のみの場合は行数を出力
          行間-Dx は 1 がデフォルト
*  オプション
          -c or -n ライン番号を表示
          -v    以外の行を出力
          -F    指定行まで出力 (指定行は含まない)
  -T    上記に追加された場合最後に行数を書く
```

```
          -R    指定行後を出力 (指定行を含)
  -i    区間指定
* 行番号 の部分が文字列 (値が 0) または -S が先に指定された
          ときはその行から Dx はなれた 1 行を表示、
  -R をつけると Dx 行まで表示但し、デフォルト Dx=1 の場合は残り全体
                但し Dx=999 の場合は一つ前
```

8.4 「fromline」アギュメントが行にあった場合、それ以後を出力

```
**************fromline***************Ver 010928=>0306***********
*機 能: アギュメントが行にあった場合、それ以後を出力
*形 式: fromeline 検索語  [入力ファイル名]  [出力ファイル名]
*形式 1: fromeline fromthis ファイル名
* -v    アギュメントが行にあった場合、それ以前を行出力
*       fromeline -v untilthis ファイル名
* -t    アギュメントから -t の後までの間を出力
*       fromline fromthis -ttothis [ファイル名]
* -C NULL 行 以後出力
*形式 2: fromeline fromthis ファイル名  出力ファイル名
*       全体を標準出力し、検索語以後をファイル出力
* -1    全体を標準出力し、検索語がある最初の行をファイル出力
************************************************************
```

8.5 「join」二つのテーブルの突き合わせ、追加

```
*************join***********Ver990605-0311******
*  機  能  : 二つのテーブルの突き合わせ、追加
* 利用法   : join ファイル 1(n 項) ファイル 2(2 項) ..... ファイル 1 が長いもの
*          : ファイル 1 の項目の直後にファイル 2 のデータが挿入される。
*          : データは 項目 ファイル 2(1 項) ファイル 1(n-1 項) となる
*両方のファイルはソートされており、スペース区切り (FS) の必要がある。
*オプション  -l 突き合わせ後の表の長さ制限 (項目を除く長さ デフォルト 50)
*            -f  ... ソートされていないファイルを横に (後ろに全体を) つなぐ
*                読み込み順に 強制的に繋ぐ
*            -F: フィルドセパレータ
*****************************************
```

8.6 「filo」全行の逆出力

```
*************cpkeiji***********Ver0212******
*  機  能  : 全行の逆出力
* 利用法   : [対象ファイル]
*オプション:
* 使 用 例   :filo *****
```

```
*****************************************
```

8.7 「findfs」フィルドセパレータを見つける

```
************findfs**********Ver020707******
*  機  能  :  フィルドセパレータを見つける
* 利用法   :findfs [対象ファイル]  ファイル指定が無い場合は標準入力
*オプション:-d  デバックモード
* 使 用 例 :findfs  /etc/passwd  ..............  : を返す。
            A=‘findfs /etc/hosts‘  ..............  スペースを返す
            asc$ "$A"              ..............  32 を返す
```

8.8 「select」ファイルのから条件に会うものを表示列に従って表示。

```
*形  式:select(020827) 表示列 (from) ファイル (where) 条件
*機  能:引数 2 のファイルのから条件に会うものを表示列に従って表示。
*使用例:select 1,2 (from) table (where) 1=test
*      :select 1,2 (from) -    (where) 1=test      標準入力を処理
*      :select 3,2 (from) file (where) 1=test%     前方一致検索
*      :select  #  (from) file (where) 1=%test     後方一致検索
*      :select  %  (from) file (where) 1=%test%    含む検索
*      :select  %  (from) file (where) 1<10        10 より小さい時
*      :select  %  (from) file (where) 1<10%       文字列比較
*      :-F     フィルドセパレータを指定 (デフォルト自動)
*      :-o     以後オプション設定禁止 (-が付いた文字列操作)
*      :-h     この案内          *      :-d     デバックモード
例:/uap/BC/select % from /etc/passwd where 1=root -F:
......   root:x:0:1:Super-User:/:/sbin/sh:::::::::::::::::::::: が返る
例:/uap/BC/select 1,5 /etc/passwd 1=root
......  root:Super-User       が返る
```

8.9 「update」ファイルのから条件に会うものを更新する。

```
*形  式:update(0210->0304->0306) ファイル (set) 変更条件 (where) 検索条件
*機  能:引数 2 のファイルのから条件に会うものを更新する。項目数 (<40)
*使用例:update table (set) 2=test,3=aaaa (where) 1=test
*      :update -     (set) 2=test  標準入力を無条件処理
*      :-F     フィルドセパレータを指定
*      :-b     バックアップファイルをつくらない
*      :-t 上記に加えファイルに書かずに標準出力にする。
*      :-f     変更条件と検索条件をファイルから読む
*      :-o     以後オプション設定禁止 (-が付いた文字列操作)
*      :-h     この案内          *      :-d     デバックモード
*バックアップファイルが/tmp/update に作られる。
```

8.10 「delete」ファイルのから条件に会うものを削除。

```
*形　式:delete(020816) (from) ファイル (where) 条件
*機　能:引数２のファイルのから条件に会うものを削除。
*使用例:delete (from) table (where) 1=test
*       :delete (from) -    (where) 1=test 標準入力を処理
*       :delete (from) file (where) 1=test 前方一致検索
*       :delete (from) file (where) 1=test 後方一致検索
*       :delete (from) file (where) 1=test 含む検索
*       :-F      フィルドセパレータを指定
*       :-o      以後オプション設定禁止 (-が付いた文字列操作)
*       :-h      この案内          *       :-d      デバックモード
```

8.11 「count」同じ行が何行あるか調べる

```
*************count********ver990824*************
*  機　能     : 同じ行が何行あるか調べる
 オプション   : -c  既に集計された [項目 個数] のファイルを更に集計
              : -t  tex format (デフォルト)
              : -k   後ろの\\を書かない
              : -r   カウント数を最後に書く
     : -s[n]    1[n]  カウントのものは出力しない
              : -n   カウント数を書かない
              : -l200      200(9000) に達したら項目を整理する。
              : -d      debag 新項目を表示
* 使 用 例   : cat file | count or count file
* 関連       : csl
*********************************************
例   : tail -6  /var/adm/messages | count
   2 & Jan 24 02:30:00 cp01 adm: [ID 702911 daemon.error]  \\
   2 & Jan 24 02:30:00 cp01 last message repeated 4 times \\
   1 & Jan 24 02:30:00 cp01 adm: [ID 702911 daemon.error] **********  SYSTEM ACCOUNTING STARTE
   1 & Jan 24 02:30:07 cp01 adm: [ID 702911 daemon.error] ********** SYSTEM ACCOUNTING COMPLETI
   を得る
```

8.12 「colcat」複数のファイルを各行を無条件に一列に連結

```
*************colcat[011025]*********************
*  機　能     :複数のファイルを各行を無条件に一列に連結
*             短いファイルに長さになる。
* 使 用 例   :colcat file1 file2 file3
*オプション  : -F     .... 連結子の指定 (t はタブ)
*関連        : join,mline
```

```
*****************************************
```

8.13 「mline」改行だけの行まで、あるいは指定行数を一行につなげる。

```
***[mline(0207)]***options(-help,-lines=,-LINES=,FS=,LS=,-d,-dbf)*
*  改行だけの行まで、あるいは指定行数を一行につなげる。      *
*   of=出力ファイルの指定                                   *
*  -LINES=10 なら 10 行連結出力                              *
*  -lines=20 なら行番号ありで 20 行連結出力                  *
*  -man   (d)elete D (c)onect その他= RET をつけて出力 *
*  sepr=セパレータの指定 セパレータが現れたら改行する *
*  llm=接続最大長 (80) 20〜200                               *
*  -tex   tex2txt の出力*を取ってつなげる (-TEX 後ろの*)*
*  -sp space1 文字の行は省く                                 *
*  FS=: 連結コードの変更    デフォルト :                    *
*  -FS  連結コードなし                                       *
*  LS=2 連結後の空白行       デフォルト 0                    *
*  -dbf なら行番号ありで出力                                 *
```

8.14 「cutter」一行の長さを制限する

-
-

```
************cutter***********Ver021107******
*  機  能  : 一行の長さを制限する
* 利用法   : cutter 対象ファイル
*オプション: -l 長さ指定  デフォルトは 80 文字
*          : -t 繋ぎにいれるスペース
*          : -d デバック (DIM=4096)
*          : -M 0 以下の文字で LF
* 応用例   : mline -FS file | cutter で行長調整
*****************************************
```

8.15 「cmpline」連続行のマスク文字列を比較し等しい時は swap に変換

```
************cmpline...ver 990903->0306*********************
*  機  能  : 連続行のマスク文字列を比較し等しい時は swap に変換
*  形  式  :cmpline 文字列のマスク ファイル 変換文字 (swap)
*      例  :cmpline  ********************* -   "        " (swap)
*           * の部分が前の行と比較され等しい時は swap に変換
*オプション: -j 連結モード * の部分が前の行と比較され
*          :     等しい時は等しい部分を除いて連結
*          : -i 対象ファイル
```

```
* -D        : ファイル１ から ファイル２ のデータを除去 (ファイルの差分)
*           : -d デバックモード
*****************************************************
```

8.16 「page」指定ページの出力

-
-

```
**************page***************Ver 040325***********
*機 能: 指定ページの出力
*形 式: page  ページ番号  [ファイル名]
*形 式: ページは -1-,--1--,1--- 等で指定されていること。
*         -は少なくとも 2 個必要
*      :-d  デバック
*********************************************************
```

8.17 「uniq」同じデータを出力しない。(sort の必要なし)

```
*************uniq***********Ver040913******************
#- 機  能    : 同じデータを出力しない。 256 文字 x2000 行
#-利用法     : uniq [ファイル]    ファイル指定が無ければ標準入力
#-オプション:-c 数字カラム指定
*            :-F  区切り文字を指定（デフォルト|）
#            :-d デバック
*********************************************************
```

9 その他

9.1 「pause」条件にあうとき, コメント出力し入力待ちとなる。

```
*************pause***********Ver5002<-981213******************
#-機  能 :  [条件にあうとき] コメント出力し入力待ちとなる。
#-利用法 :  [コメント]  [when]  [条件]
#-返り値 : n,N  -->2
#-              y,Y  -->0
#-              others->1
********* -d=デバック*********************************
```

9.2 「for」繰り返し 制御変数は#に格納

```
*************for***********Ver021010*********************************
*  機  能  : 繰り返し       制御変数は\#に格納
* 利用法   : for [オプション] 1,10,[1]  CMD  初期値、終了値、ステップ  コマンド
```

```
*オプション: -F 実数モード  -s 秒  sleep_time の指定  最初に指定
* 使 用 例 :for 1,10  ping 133.63.56.\#
************************************************************
使用例 1 :/uap/BC/for 1,10  printf \#   ....... 12345678910 を返す
使用例 2 :/uap/BC/for 1,10,2  Let 10+\#   ....... 11 13 15 17 19 を返す
```

9.3 「TGMk」テラ (T) ギガ (G) メガ (M) キロ (k) に単位変換。

```
************TGMk********0409************
* 形 式:TGMk 数字  [閾値:1.0]
* 機 能:引数 1 をテラ (T)G(ギガ)M(メガ)k(キロ) 単位をつける。
*使用例:TGMk 10000            10k ......
*オプション: -f 出力フォーマット [デフォルト%g]  例 -f   0.00
************************************************
```

9.4 「data_form」データ形式解析

```
*************data_form***********Ver04Feb18/a**********************
#- 機  能   : データを解析し 英数字 a       数字 i         スペース s
#-                            実数は I  セパレータ [,;:+-/|] S    その他は A
#-利用法    : data_form  04Feb18    ---->i2 a3 i2 を返す
#-オプション: -v 数字は 0 asc は a 記号はそのまま その他は A  04Feb18 -->00aaa00
#-            -s 数字のみは 0、英数字 1 EUC 漢字が含まれるときは 2 数字のみは 0、英数
字 1 EUC 漢字が含まれるときは 2
#-            -x ヘキサの場合は 0 それ以外はもとの文字
#-            -d デバック
*************************************************************************
利用例 1  英数漢字の判断
FM=‘/uap/BC/data_form -s  $1‘
case $FM in
        0)      W="pid=$1" ;; #--数字のみ
        1)      W="acct='$1' ;; #-英数字
        2)      W="name='$1'" ;; #-漢字混じり
esac
利用例 2  日付の形式判断
FM=‘/uap/BC/data_form -v  $1‘
case $FM in
        0000-00-00|0000-0-00|0000-0-0)     #--年-月-日
        00-00-0000|0-00-0000|0-0-0000)     #--月-日-年
        00aaa00)    #-04Feb18
esac
利用例 3 16 進判断
data_form -x 12:fe:e0    ............00:00:00 を返す
```

9.5 「Knjcount」引数 1 の文字列の漢字数を出力。

data_form コマンドの方が有効なため現在ではあまり使われていない。

```
*       Knjcount(990224)        *
* 形 式:Knjcount 文字列
* 機 能:引数 1 の文字列の漢字数を出力。
*使用例:Knjcount 'test 漢字数'
       :Knjcount  -     : 標準入力を対象
       :Knjcount  -  2 : 漢字が 2 文字以上なら全体を出力
       :Knjcount  -n    :以後オプション判定しない。
```

9.6 「data_form」データ形式解析

```
*************data_form***********Ver04Feb18/a**********************
#- 機  能   : データを解析し 英数字 a       数字 i         スペース s
#-                              実数は I  セパレータ [,;:+-/|] S     その他は A
#-利用法     : data_form  04Feb18    ---->i2 a3 i2 を返す
#-オプション: -v 数字は 0 asc は a 記号はそのまま その他は A  04Feb18 -->00aaa00
#-            -s 数字のみは 0、英数字 1 EUC 漢字が含まれるときは 2 数字のみは 0、英数
字 1 EUC 漢字が含まれるときは 2
#-            -x ヘキサの場合は 0 それ以外はもとの文字
#-            -d デバック
*****************************************************************************
利用例 1  英数漢字の判断
FM=‘/uap/BC/data_form -s  $1‘
case $FM in
        0)      W="pid=$1" ;; #--数字のみ
        1)      W="acct='$1' ;; #-英数字
        2)      W="name='$1'" ;; #-漢字混じり
esac
利用例 2  日付の形式判断
FM=‘/uap/BC/data_form -v  $1‘
case $FM in
        0000-00-00|0000-0-00|0000-0-0)     #--年-月-日
        00-00-0000|0-00-0000|0-0-0000)     #--月-日-年
        00aaa00)    #-04Feb18
esac
利用例 3 16 進判断
data_form -x 12:fe:e0    ............00:00:00 を返す
```

9.7 「crypt」文字列と key を使って文字列を暗号化する。

```
*************crypt***********Ver020708A******
*  機  能  :文字列と key を使って文字列を暗号化する。
```

```
*  形式     : crypt 暗号化する文字  salt(2 文字)
*オプション: -l 変更日も出力 (shadow 用)
*           : -r salt 無しで暗号化
*****************************************
例:/uap/BC/crypt  aaaa a -l ........aaplnuZecTgLA:12816 を返す
∵:12816 は 1970 からの日数
   no_of_days 1970-1-1 today  ......12816 を返す
```

9.8 「nensort」日付でソート

```
************nensort***********Ver991201******
*  機  能  :99 から 70 は -1 から -30 としてソートする。
* 利用法   : 対象ファイル
*オプション:-n2    .... 何番目のワードか指定
*           :-F.    ..... セパレータの指定
*           :-d     ..... デバックモード
* 使 用 例 :nensort file オプション
*****************************************
```

10 ベーシックライクコマンドの組み込み

/uap/BC/src および /uap/BC/src/BL にソースファイルと makefile が収められている。また、ftp.nirs.go.jp/pub/usr/hongo/の下に tar ファイル『BC.tar』があるこれらを適当なディレクトリーにコピーし、tar ファイル の場合は tar -xvf BC.tar で展開する。makefile 中 の
TD= /uap/BC
をインストールするディレクトリーに変更し、make all を実行すると ベーシックライクコマンド が 組み込まれる。但し、asc$,bin$,chr$,hex$,mid$,right$,string$ はそれぞれ ascS binS chrS,hexS,midS,rightS,stringS としてコンパイルされるのでこれらをリネームしなければならない。

11 ベーシックライクコマンドのサブプログラム

ベーシックライクコマンドはサブプログラムをインクルードして作られている。これらのサブプログラムをインクルードすることにより、容易に新たなコマンドを作成することが出来る。

```
AutoScale.c:#-  関 数 名: AutoScale(axis,mode)
           :#-  機    能: 自動スケール設定
           :#-  引    数: axis=0 X 軸  axis=1 Y 軸 mode=0 自動 mode=1 リニア mode>2 ログ
           :#-  作    成: Date 2001/04/14 s_hongo@nirs.go.jp
           :#-gloval  min[axis]=最小値、max[axis]=最大値、mean[axis]=平均値 は軸に 1 個
           :#-        momnet[axis][]=積率は y 軸に対して yno 個ある。
DateOut.c:/*#--- BL/DateOut(struct tm *T, char *fmt) 日付の指定形式出力
         :  #- fmt は ymdwhMs (年月日時分秒) で指定する。その他はそのまま出力
GetOutputForm.c:/*#- GetOutputForm(FS,CM) tex 表等を読んでカラムの長さ (fm) を求める
               :  #-define ROW 48
               :  #-global char   buff[NO_LINE],
```

```
              :  #-       char   fm[ROW][32];  出力フォーマット
              :  #-       int eod,
              :  #-       FS=& セパレータ   CM=%... はコメント扱い
OutputTbl.c:/*#---OutputTbl(char *s,char *Fs,char CM,char *Head,char *Tail)----------
           :  #- tex フォマットの s を 出力
           :  #- global char fm[][]; int ROW
           :  #---------------------------------*/
Read2dd.c:*#- 関 数 名 : Read2dd(FILE fi,int minyear,int minmonth)
         :*#- 機    能 : 日付＋数字データの読み込み 日付は積算日に換算される。
         :*#- 呼出形式 : int Read2dd(*fi,*minyear,*minmonth )
         :*#- 引    数 : *fi; ファイルポインター (stdin) *minyear,*minmonth インテジャポイ
ンター
         :*#-  帰り値  :  nod No of Data
         :*#---Grobal----- 2dread.h  X[].Y[]
         :#-include "BL/dellf"
Read2dt.c: if (dbf) printf("#--負のときは自動加算しない= %d (%d)\n",month,*minday);
         : if (adc >2 && month>0) {fprintf(stderr,"##-日付の順序をチェックして下さい。-\n"); month=0;}
Read2td.c:#-include "BL/dellf.c"
TestTblForm.c:/*#---TestTblForm() 入力表ファイルのフォマット検査 (global buff[],eod)-- */
Word:/*#- Word( s, separater, wordno)
    :  #-   1 ワードの切出 0x0a はいつもフィルドセパレータとする
    :  #- s 文字列       .... 入力、出力 (代えられる)
    :  #- separater セパレータ
    :  #- wordno 番号
WordTest.c:/*#---------lib/WordTest(char *s,int func)-----------------------------
          :  #-文字列比較をし、TestWord の何番目にあるかを挙げる。 無いときは 1000
          :  #-func =0 完全一致 func=1 前方一致  func=2 s に=があると前方一致
          :  #-global char TestWord[][]; int WordEnd
          :  #-----------------------------------------------------------------*/
adjustword.c:#-adjustword(int mod,char *s,int length) s を length の長さに調整する
            :#- s は 十分な長さをもつこと
            :#-
            : if ( l >length ) { /*#- 長さがオーバーの時 */
            : if (mod==0 ) return(-1) ;  /*#- そのまま           */
            : *(s+length+1)=0 ;return(-1) ;  /*#- 強制的に切る       */
            : if (mod<1 || mod>2 ) {  /*#-自動判定  数字で始まれば 前 それ以外は後ろ*/
            : case 1: /*#-前にスペース*/
            : case 2: /*#-後ろにスペース*/
            : default: /*#-自動判定  数字で始まれば 前 それ以外は後ろ*/
chkdatatype.c:/*#-chkdatatype(char *s) データの型を調べる
             :  #-0- 99 数字型
             :  #-      0       数字のみ
             :  #-      1 -9    10 以内のスペースを含む数字 (,)
             :  #- 5    3 桁, 付き数字
             :  #- 8       上記外
             :  #- 10      10 個以上のスペースを含む数字
             :  #- 20      セパレータ;:を含む数字
             :  #-100-199 文字型
             :  #- 100  セパレータ;: を含まない文字列
             :  #-200-299 文字型+数字
             :  #- 200 セパレータ;: を含む文字列
             :  #-300-399 ????
             :  #-#include "/uhome6/s_hongo/src/BC/BL/dellf.c"
chkdateform: #---chkdateform(s) date format のチェック
           : #-grobal yaer,month,day,dbf,ymod:0=年-月-日 1=月-日-年
```

```
           :        /*#-------12-07-1995 ---or 2000-01-02--*/
chrswap.c: #- 関数名:chrwap(*t,*s,d)
          : #- 機　能:対象文字列 t の中の s に表れる文字はすべて文字 d に変える
          : #- 注　意:t が直接変えられる。
date_sub.c:#---date_sub(y,m) 年、月の day_of_manth, 年始めからの日数、曜日を求める。
dellf.c:/*#-dellf(s)  最後の LF 等 0x0d,0x0a,0xb 0xc を取る
        : *#-ポインタ渡しのため呼出側で十分なデイメンションをとること*
        : *#-include "/uap/src/lib/       "
        : *#-char *strchr() を定義すること。
delsp.c: #- 関数名:delsp.c
        : #- delsp(モード, 文字列ポインター) 注:文字列は変えられる
        :      case 0 : /*#- default 後の半角スペースを削除 */
        :      case 1 : /*#- -a 全てのスペースを奪る。 漢字スペースも含む*/
        :     case 2 : /*#- -f フロントスペースを奪る*/
        :     case 3: /*#- -c 連続したスペースは一個に*/
        :     case 4: /*#- -t 後の半角全角スペースを取る */
dfa.c:/*#-char *dfc(char *s) データの型を調べる Data Format Anarizer
      :  #-グローバル変数 DFA に出力 a3 S2 i6 等
      :  #-i 整数型    I  数字
      :  #-a 文字型    A  漢字
      :  #-s スペース  S  セパレータ;:
      :  #-例:abc;;123456 は  a3 S2 i6 となる
dfa0.c:/*#-char *dfc0(char *s) データの型を調べる Data Format Anarizer 0
       :  #-出力 aa00;-+ 等 自分自身 (char *s) を変えるのでコピーして渡す。
       :  #-0 数字は             0  asc   は a に
       :  #-EUC 漢字第 1 バイト目は b  EUC 漢字第 2 バイト c に
       :  #- !""#$%&''()*+,-./:;<=>?[\]^_{|}~. はそのまま
       :  #-その他 は A に
       :  #-関連 include "BL/dellf.c"
divider.sub:/*#- divider.sub 指定分に整数分割する t=全体長、d=分割数 o に出力*/
file_open.c:#---注釈------
find_out:#- data[] から keyword を検索する。
        :#-  swf=1 一致検索 uDB では keyword には=まで入る。
        :#-  swf=2  前方一致検索
        :#-  swf=3  曖昧検索
        :#-  swf=4  変更 挿入
        :#-  swf=5  カウント
        :#-  swf=6 一致検索で gloval FS 後の gloval *gp のみかえす。
        :#-  swf=7 削除
        :#-  swf=8  add1
        :#-  リターンコード 1 データ save
        : if (dbf) printf("#-----         swf=%d keyword=%s-------\n",swf,keyword);
        : if (dbf) printf("#--add1 keyword=[%s] ---\n",keyword);
fpath_open.c:/*#----- FILE *fpath_open(char *path,char *fname, char *comment):
            :  #-    path の中のファイルのオープン ----
            :  #-      comment は失敗した時に表示され exit する。
            :#-例題
            :#-        fr=fpath_open(char *path,char *fname,char *comment)
            :#-----------------------------------------------------------*/
getYdata:#-------------BL/getYdata from s --X[500],Y[500][yno] の Y を得る (関数ではない)
        :#- char *a;  double y,ymax,ymin; double mean[31]; int j,y,yno,
        :#- included  Read2dt.c Read2td.c
getymd.c:#---日付のデータを得る--int getymd(char *s,int *year,int *month,int *day)--
        :#- s -12-07-1995 ---or 2000-01-02
insert_space.c:/*#-char *insert_space(char *p,int no_of_space) global dbf
```

```
             :  #- p の位置に no_of_space のスペースを入れる #include "BL/              "
             :  #-"abcdef" ポインタが c をさしていれば ab    cdef を返しポンターは c を指す
instr.c:#-instr(char *ssr, char *fsr ,int start )  include "BL/instr.c"
       :#-mod が 10 以上なら 小文字列変換して比較
       :#-global int mod,dbf
jump2FS.c:/*#----char *jump2FS(char *q,char FS) FS まで ポインター q を進める ----*/
         :/*#---"から"までは 1 区切りとする-- #include "BL/         " ---*/
list.sh:#- list.sh
       : grep "#-" $FILE |grep -v grep
matMp.c:/*#- 指定文字列のマトリックス ポインタを作る
       :  #- int matsize matMp(char *source) source 文字列は変えられる
       :  #- include BL/Mp.h
       :  #-   char *Mp[MPX][MPY] グロバルポインター matsize=100*tate+yoko
matMp2.c:/*#- 指定文字列のマトリックス ポインタを作る
        :  #- int matMp2(char *source,char FS1,char FS2) source 文字列は変えられる
        :  #- include BL/Mp.h
        :  #-   char *Mp[MPX][MPY] グロバルポインター return=100*tate+yoko
midS.c:  #-char *midS(s,start,end) s 文字列から start end を切り出す s は変えられる。
      :  #-  start <0  指定 end <1 により、
      :  #-       文字列指定のカットにも対応 startstr[64],endstr[64] に入れる
      :  #-注意 Global で以下の定義が必要
      :  #-  char *midS(),startstr[64],endstr[64],*strstr();
mtterm.c:#-関数名:mtterm.c  複数項の算術計算
        :#- pp はプログラムポインターで global
        :#- acl アキュムレータレベル 括弧のたびに深くなる
        :#- 単項の処理 sterm()  */
mtterm.h:#-------mtterm.h-----mtterm.c のヘッダー 011013 (に対応)----------
nengo2sub.c:#-  年号を西暦に変換 nengo2sub(char *s)
           :#-  西暦*100 + データ番号 を返す
no_of_days.c:#----西暦 0 年からの日数を求める。-----------
            :#- 引数：年、月 (月が負のときはその月の日数)
printmat.c:/*#-----printmat(int サイズ,int 出力形式)-行列出力 subroutine-------------------------
          : *#-  include "BL/          "
          : *#- a[i][j] はグローバルで入力も出力もここが用いられる。
          :  case 0: /*#- g 変換で行はスペース区切り */
          :  case 1: /*#- g 変換で 行は LF */
          :  case 2: /*#- 6g 変換 行は LF */
          :         case 3:  /*#- default matprint  12g 行は LF*/
          :         case 4: /*#- 6.3f 変換 行は LF */
          :  case 5: /*#-記号 1 文字表示 スペース区切り 行は LF */
          :      default: /*#- 12.2f 表示 行は LF */
read2buff.c:/*#-  read2buff(int mod)     buff に 読み込む 完了 0  未完 mod
           :  #-  制限行 NO_LINE (1024)
           :  #-global char   buff[NO_LINE][512], int eod
rstr.c:  #- rstr(char *s) ASCC 文字列のヒックリ返し
      :  #-      char *s の並びが逆になる ASC のみ対応
s2nengo.c:#- BL/s2nengo.c  西暦をあたえて 年号の番号* 1000+年 を得る
scaleoption:#---------------BL/scaleoption=int checkoption(s)--------
           :#-xstart=","xend=","xmin=","xmax=","xlog=",
           :#-"ystart=","yend=","ymin=","ymax=","ylog=",
           :#-"mark=","msize=","csize=","lwidth=",
           :#- "plottype=","if=","of=" ,"debf=",
           :#- "xtitle=","ytitle=","Title="
shifter.sub:#- BL/shifter.sub 文字 (ch) のシフト (shifter) 関数にあらず
           :#- 変数 ch,shifter,dbf,AU=0x7E
```

```
str2word.c:/*#- int str2word(s,fs) 文字列 s を分解して W[][] に
          :  #-#define VT 32
          :  #-#defile VY 128
          :  #-global W[VT][VY]
str2word_pointer.c:/*#- int str2word_pointer(s,fs) 文字列 s を分解して *P[] に
                 :  #-#define VY 32
                 :  #-global char *P[VY]
strmask.c:#-int strmask(int mod,char *s,char *mask) マスク文字列の取り出し
         :#- mod=0 mask の ****** に相当する文字を s に返す
         :#- mod=1 mask の ****** の部を除いた文字を s に返す
strmatch.c:/*#-************************************************************
          :  #- 関数名:strmatch(*t,*s) 見つかった場合は 正 (1-) 無い場合は -1
          :  #- 機　能:対象文字列 t の中に s の先頭文字列があるか
          :  #- 注　意:文字列は双方ともスペース区切り s は =も区切りとする。
          :  #- int *strcmp() ; を宣言すること
          :  #- 040817 041117************************************************
          : if (dbf) printf("#-strmatch テストワード [%s]\n 順 キー 差分\n--------------\n",w);
          : if (dbf) printf("#-strmatch--- 終了---- j=%d\n",j);
testtblform.c:/*#--------入力表データのフォマット検査------testtblform()-------
             :  #- gloval buff[],eod
             : if ( sql > csv || sql >j )  return(5); /*#- sql */
             : if ( csv > 2*j )     return(4);  /*#- csv */
             : if ( html > 2*tex )  return(3);  /*#- html */
             : if ( tex  > 2*html ) return(2); /*#- tex */
timeconv.sub:#-timeconv(char *s) s は入出力に用いられる。
            :#-************%s***********Ver%s*********************\n\
            :#-  機　能  :  時間の変換  03:18:20 --->(03*60)+18*60+20 \n\
            :#-          :  5 時 30 分 20 秒 ----->19820\n\
            :#-          :  自動判定 \n\
            :#- 利用法   :%s  12:30 ...>秒に変換 \n\
            :#-          :%s  230   ...> 3:50 に変換 \n\
            :#-************************************************************\n
vecStrp.c:/*#- int size =vecStrp(*文字列, フィルドセパレータ, ポインタの配列のポインタ)
         :  #- 指定文字列のベクトル ポインタを作る        文字列は変えられる
         :  #- int vecStrp(char *source,char FS,char *vp[]);
         :  #- char svec[1024],*vp[64],FS; vecStrp(svec,',',&vp[0]);
vecprint.c:/*#-vecprint(int mod,char *s,char FS) ベクトル (1 行) の出力
          :  #- mod は 0 が tab 出力 8 が HTML 出力
```