研究ノート

# コンピュータ教室での出欠点検用プログラム

## ——構文処理言語 awk による試作例——

鹿子木 幹雄

（受付　2000年10月10日）

以下は，本学情報センターの Compaq Tru64 UNIX V5.0A（Rev. 1094）に標準装備されている構文処理言語（pattern scanning and processing language）「オーク（awk）」XCU5.0版によって作製したコンピュータ教室授業の出欠点検プログラムの試作報告である。

## はじめに

本プログラム試作の目的は，当初は出欠点検の自動化という便宜にだけあった。しかしながら，このようなプログラム製作が，同時に，履修学生に対するある種の教育効果をともなうものであることも指摘しておきたい。それは，学生に，授業というひとつの現実に対応する既成のプログラムがないので，それを作製する必要性があることを示すこと，プログラム作製のためには現場の調査・取材が前提になることを例証すること，あえてつけ加えれば，そのプログラム作製を1文系教員が遂行したことを例示することである。

いわゆる情報関連技術（Information Technology）の進化・普及は，報告者が担当する授業の履修学生に関するかぎり，かなりの疎外感や焦燥感をあたえているように見える。それは端的には，今春の「情報化社会と人間」授業の5倍という抽選率や，歴史関連授業の質問紙への「コンピュータを買うべきか」という記載などのかたちで現れている[1)]。極論すれば，『蜘蛛の糸』（芥川龍之介）の強迫心理さえかいま見えるのである。報告者は「E-ジャパン構想」などの語がとび交う今次国会の補正予算論議などがこのような状況を一層悪化させて行くことをおそれている。

本学部の「コンピュータ入門」は基本汎用ソフトウェア（WORD, EXCEL, PowerPoint等）の体験・習得を目標にしているが，これでは，他学部の初心者クラス同様，ソフトウェアの

1) 教養ゼミ履修者6名の半数が自分のコンピュータをもっているが，使用状況は聞いていない。講義受講者・コンピュータ実習履修者についてこの種のアンケートをしたことはなく，その実態は不明である。本学のカリキュラム改善のためにも，学生の情報技術の利用状況に関する全学的なアンケート調査の実施をすることが必要であろう。

受動的なユーザーを育成することに終わってしまいかねない。報告者にはコンピュータ教育の体系を論ずる資格はないが，個々の学生に情報関連技術（IT）の空隙（niche）を実感させて，なんらかのプログラム作製をさせるのが良いのではないかと思う。汎用ソフトウェアの総合的な習熟などよりも，各人のホームページづくりの方に意味があるのかも知れない。

awk は，構文（pattern）の検出（matching）に対応する処理（action）をするコンピュータ言語であり，文の構造については，項目（field）とその区分（delimitter/separator）を基礎概念としている。プログラムは，処理群（rule）ごとに説明されるのがふつうである。しかし，報告者はこのような原語をカタカナ書きした「専門語」の使用には賛成できない。いたずらに原語のカタカナ書きをすることは，タコツボ型「専門家」をつくりだすだけだからである。そこで，以下ではできるだけ「日本語」を使って報告文を書く。

## 教室の環境

まずプログラム製作が可能であるかどうかを情報センターに照会した。すると，"last" という命令が UNIX にあることが分かった。この命令は，コンピュータ教室の利用者が，画面に配置されているアイコン群の中から "TeraTerm" を選択し，情報センターの UNIX に接続した場合に残される記録ファイルを呼びだし，画面に表示するものである。この表示順は現在からファイルの開始時点まで遡及するようになっている。同じ目的でホームページを開設し，それへの接続を検知して出席を取るプログラムを製作することも考えたが，これは接続機器（TCP/IP）を検知できるだけで，使用者を特定することはできないらしい[2)]。

"last" 命令でえられるファイルの記録形式は，awk の表現で言えば空白（スペース）を区分文字（セパレーター）とするもので，次のような 2 種類の構文になっている。

```
kanokogi  pts/2        rm549              木  9月 4日 16:30   still logged in
xxxxxxx   pts/6        pcx18.b1.shudo-u   木  8月31日 15:05 - 16:03  (00:57)
```

すなわち，各人の接続記録は 1 行で，ログイン名・接続ホスト関連・接続場所関連・曜・月日・開始時刻－終了時刻・(接続時間)，または使用中であるむねが表示されるが，そのうちの教室関連の項目では，最初の 3 字で教室別（FPC, PC1, …PC7…）が判明し，次の 2 字で席番号 01…30 など（教卓は 99）が識別されるようになっている。ただし，月日の項は日付が 1

2) もともとは出席点検のために開設した http://alpha.shudo-u.ac.jp/~kanokogi/ は，その後，情報技術に関する学生の意見を反映する場になっている。http://ns1.shudo-u.ac.jp/~kanokogi/ にも対応するホームページがある。両者は，すでに本学部ホームページからの接続（リンク）が可能になっている。

桁の場合には上記の「9月 4日」のように月と日の間に空白が入るため，awk 処理の場合に，べつべつの２項目として扱わなければならなくなる点に注意を要する。

## プログラムの製作過程

このように，情報センターのUNIXに接続記録がある以上，"last"のデータと履修学生データとを結びつければ良い。この２つのデータに共通する項目はないから，このままでは比較できないが，ログイン名は"finger"で照会できるから，これを照合項目として履修学生データに加える。こうして以下のように３段階（３種類）のプログラムを製作した。

①作業は単純なものから始めた。それは，２群のデータの照合結果を履修者名簿順に作製し，それをその都度印刷して利用・保管するという単票形式のものであった。この出力は，次回に出席を取ったときに上書きされてしまうから，印刷された単票をきちんと保存しておかなくてはならないことになる。

②そこで次に，毎回の出欠点検結果を単票印刷する点までは同じだが，結果を後日識別しやすい「年月日時限」名のファイルに保存しておいて単票紛失に備えることにし，蓄積されるファイルを統合するプログラムの製作は，他日の余人の労を待とうと考えた。そこで，これで報告者なりのプログラム製作は終わったと考え，一旦報告文を書きおえた。

③しかし考えてみれば，上の作業でファイル名として利用した「年月日」は，そのまま見出し項目として使えそうだ。報告者は，既成のソフトウェアのような可塑的な構造をもったものをデータベースソフトと思いこんでいたが，一覧表タイプの出力がえられればそれで一応満足できるではないか。授業は継続的なものだから，履修者の出席状況を見て，前回欠席者に注意をうながすことなどは必要で，それには一覧表タイプが良い。

３段階のプログラムのうち，①②のレベルで製作したプログラムにも，バックアップ用としては依然多少の利用価値がある。最後の③が本報告の簡易／疑似データベース型のプログラムである。この設計段階で，記入欄の余白部にゆとりを持たせるため，履修者のローマ字表記の項と，UNIXログイン中またはUNIXからのログアウト時刻を表示する項の２項目を省略した。その原型を，後期授業の最初の授業から使用しはじめ，途中なんどか修正・改善して現在にいたった。

おもな修正・改善点はふたつある。ひとつは，実際にこのプログラムを第１週の授業で使用したとき，「出席を取りなおす」必要が生じ，そのたびに，出席合計時間数を重複して加算してしまうことが分かったことである。これは，当日の出欠データが確定するまで，何度「出席取りなおし」をしても，履修者名簿ファイルの本体を書きかえず，次週の出席を取るときに初めて，名簿ファイルの本体を書きかえるように原プログラムを変更することで解決した。

もうひとつは，出欠データの記入欄の制約の問題で，テストファイルによって，8週間以上の表示の場合「折りかえし」が生じることを確認した後，「折りかえし」回避のため，最終7週分だけを画面表示するように原プログラムを変更した。

## 使用言語 awk についての「発見」

ここに記すことがらは専門家からは失笑を買うと思う。しかし，1冊の日本語版説明書[3]の独学でプログラムを試作した報告者には，それなりの試行錯誤と「収穫」があったので，そのメモを以下に記しておく。

①連想配列の存否を調べる条件文"if(key in array)"は，情報センターの awk では使えず，エラー表示になやまされた。したがって，"if(array[key] != "")"の条件文を考案し，それで代用せざるをえなかった。

②初期2段階のプログラムでは，長いログイン名と短いログイン名の表示のずれに悩まされ，それを処理する条件文を書いたが，短いログイン名の末尾に空白を加えてログイン名を同長にしておくことによって，作表（タブ）文字を項目入力区分（FS）にしたとき，初期のプログラムで使っていた条件文が不要になった。

③履修者名のローマ字表記についても，その長短になやみ，上と同じ配慮から空白部をアンダーバー文字（_）でうずめたが，これもふつうの空白になおして，読みやすいものにし，読みとり時の項目区分文字（FS）を作表（タブ）文字にしておけば良かっただろう。

④このように，この目的で指定する項目入力区分（FS）・項目出力区分（OFS）は処理ファイルの構造に応じてプログラムの途中でも変更できた。例文ではこれらの区分文字がいつでもプログラム（スクリプト）の BEGIN 部にあったので，はじめ，報告者はこれらの区分文字の位置が固定されているものと誤解してしまっていた。

⑤文字関数 split() の分割文字（セパレーター）には，漢字（2バイト文字）も使うことができた。例えば，"[年月日]"，"[時分秒]"のようなものである。これで，情報センターの日本語表現による時刻データから数値部分を切りだすのが容易になった。

⑥上記②の目的で空白を加えて文字長を合わせたログイン名は，split() 関数で分割文字（セパレーター）を空白（スペース）にするだけで，配列の引数にする際，簡単に「裸」にすることができた。

⑦関数 system() 内の UNIX 命令記述には別表現（alias）は有効でなく，フルネーム表現

3) Dale Dougherty & Arnold Robbins, 福崎俊博訳『sed & awk プログラミング』改訂版，オライリー・ジャパン，1997年（情報センター所蔵）。翻訳書であるためか，報告者が雑駁な利用者だったための限界なのか，情報センターの awk XCU5.0 版や日本語（2バイト文字）での利用法とは必ずしも整合していないように思われる。

にしなければならなかった。例えば，'date + "%c %a"' の別表現として自作した 'day' や正規の 'Mail' の別表現である情報センター既成の 'mail' などは使えなかった。

⑧最後の疑似ヨコ方向スクロール部分で，"←"，"→" キーによって，データの表示範囲を選択させようとしたが，情報センターの awk XCU5.0 版が，"¥xhex" の16進数や，"¥ddd" の 8 進数による制御文字（コントロールキャラクター）の表現を受けつけないことが，試行錯誤や実験ののちに確認されたので，結局 標準化されていた後退（Backspace）文字によって代用せざるをえなくなった。

⑨また，最後まで分からなかったのが awk 文（スクリプト）の BEGIN …END という基本的な 3 部構成についてである。報告者の書いたプログラムは，なぜか ほとんど BEGIN 部分だけで終わりがちであり，どこか「鯛の兜煮」ふうで，とてもスマートな文（スクリプト）とは言えないと自嘲している。独学によるプログラム製作の限界であろう。

プログラム作製の段階でしだいに判明してきた①～⑨の「発見」を，先行する 2 プログラム[4] の細部についてすべて修正するべきだっただろうが，それぞれ一応の動作はしているし，いまさら変更するのも煩瑣なので止めた。プログラム製作の未熟さを記録しておくのは，報告者への批判と改善のための手がかりとなりうるから，それも教育上，良いことかも知れないと思っている。諸賢の批判と提案に待ちたい。

## プログラムの本文[5]

```
awk '# program file: "table" ;  text files: "intro.tbl","sogo.tbl",
     # "hist.tbl"; test file: "list.tbl" 2000.9.19～10.8 (c) kanokogi

#1 process the "date" data
BEGIN { OFS = "¥t"
        "date +¥"%c %a¥"" | getline     # date and time with week
        split($1,date,"[年月日]")       # kanji as separaters
        split($2,time,"[時分秒]")
        week = $3
```

4) 過渡的なプログラム，"class" と出力ファイル "sheet" および "check" なども，ともに報告者の /home/com/kanokogi に，複写・実行の可能なモードで保存してあるので，教職員・学生とも，これらを自由にコピーし，分析したり試用したりすることができる。

5) これも複写・実行可能なモードにしてあるので，"cp /home/com/kanokogi/table" などの指定でコピーし，#2 の部分を変更するだけで，他の授業についても自由に利用していただける。その際，原プログラムが報告者によるものであることを明記していただければ幸いである。

```
        item = date[2] date[3]          # 月日item for table
        date[2] += 0; date[3] += 0      # numeralize
        month = date[2] "月"
        day = date[3] "日"
        now = time[1]*60 + time[2]      # present time counter

#2 select the list at the day
    if ( week == "月" && now >= 650 && now <= 740 )
        { labo = "pc7"; list = "intro.tbl"; file = "intro"; bt = 650 }
    else if ( week == "月" && now >= 890 && now <= 980 )
        { labo = "pc4"; list = "sogo.tbl"; file = "sogo"; bt = 890 }
    else if ( week == "水" && now >= 890 && now <= 980 )
        { labo = "pc6"; list = "hist.tbl"; file = "hist"; bt = 890 }
    else if ( week != "月" || week != "水" || now < 650 || now > 980 )
        { labo = "pc1"; list = "list.tbl"; file = "list"; bt = 720
        month = "9月"; date[3] = 4; day = "4日" }

#3-1 read "last" data before 10th of month (NF=10)
if ( date[3] < 10 ) {
    while ("last" | getline)
    if ( substr($3,1,3) == labo && $5 == month && $6 == day)
        { s = substr($3,4,2)                    # scan seat
            if ( $9 == "logged" ) lt = now      # "still logged in"
            else { split($9,l,":"); lt = l[1]*60 + l[2] }
        if (lt > bt) { seat[$1] = s } }         # avoid overwrite
        }

#3-2 read "last" data after 10th of month (NF=9)
else {
    while ("last" | getline)
    if ( substr($3,1,3) == labo && $5 == month day )
        { s = substr($3,4,2)                    # scan seat
            if ( $8 == "logged" ) lt = now      # "still logged in"
```

```
        else { split($8,l,":"); lt = l[1]*60 + l[2] }
      if (lt > bt) { seat[$1] = s } }          # avoid overwrite
      }

#4 check the bottom of the "last" command
      print "  ¥"last¥" 開始データ → " $0     # wtmp of "last"
      if ( $4 != "8月13日" && list == "list.tbl" )
      { printf("  ¥"last¥"ファイルが更新されたので, 9月4日のテスト")
      print "結果が正しく表示されません。"
      exit }

#5 avoid multiple score & copy file to list
      { FS = "¥t" }                            # renewed FS
  while ( (getline < file) > 0 )
    if (NF == 6)
      { z = split($6,it," ")
      if ( it[z] != item ) {                   # if not same day
      system( "cp " file " " list )            # copy file to list
      close(list) }
      }

#6 print the list with seat-number or absent symbol
  while ( (getline < list) > 0 && $0 != "" )   # avoid last line "欠"
    if (NF == 1)                               # rewrite title
      { print $0 > file }
    else if (NF == 6)                          # add new item
      { print $0 " " item >> file }
    else if (NF == 4) {                        # add new data
      split($3,n," "); name = n[1]             # strip login name
      if (seat[name] != "") {                  # check seat array
        p = substr($4,1,2); ++p                # "計" increment
        q = substr($4,3)                       # former queue
        s = seat[name]                         # seat data
```

```
            if ( p >= 10)                    # write seat data
            { print $1,$2,$3,p q "  " s " " >> file }
            else print $1,$2,$3," " p q "  " s " " >> file }
        else print $0 "  欠 " >> file          # write "欠" data
        }

#7-1 display the result until 7 weeks
        close(file)
        while ( (getline < file) > 0 && $0 != "" )
    if ( NF == 1 )                           # print title
        { print }
    else if ( NF == 6 && split($6,j," ") < 9 )   # check overflow
        { print
        z = split($6,j," ") }
    else if ( NF == 4 && split($4,k," ") < 9 )   # sheat/"欠" data
        { print }

#7-2 display the latest 7 week result
    else if ( NF == 6 ) {
        printf ( "¥t¥t¥t¥t¥t" )              # print tabs
        z = split($6,j," ")
        printf j[1]                          # print"計"
        for ( i = z-6; i <= z-1; ++i )       # shift date item
                printf " " j[i]
        print " " j[z] }
    else if ( NF == 4 ) {
        printf ("%s¥t%s¥t%s¥t", $1, $2, $3)     # print fixed data
        z = split($4,k," ")
        if (k[1] >= 10) printf k[1]          # print total
        else printf " " k[1]
        for ( i = z-6; i <= z-1; ++i )       # shift seat/"欠" data
                printf "  " k[i] " "
        print "  " k[z] " " }
```

```
        close(file)
        if ( z > 8 ) {
        print "  表示始め = " j[z-6] ", 表示終り = " item
        printf("  Backspase(またはCtrl+H)キーで表示の範囲を変えられ")
        printf("ます(q to quit)。") }
        else if ( z <= 8 ) { exit }
        }

#8-1 scan keyboard input
    $0 ~ /^([qQ]|quit)$/ { exit }
    $0 != "" {
        if ( $0 == "¥b" && z+x-6-1 >= 2 )       # scan BS key
                { --x
                system ("clear")

#8-2 display the 7 week result with shifting
        close(file)
        while ( (getline < file) > 0 && $0 != "" )
    if ( NF == 1 )                              # print title
        { print }
    else if ( NF == 6 ) {
        printf ( "¥t¥t¥t¥t¥t" )                 # print tabs
        z = split($6,j," ")
        printf j[1]                             # print"計"
        for ( i = z-6+x; i <= z-1+x; ++i )       # shift date item
                printf " " j[i]
        print " " j[z+x] }
    else if ( NF == 4 ) {
        printf ("%s¥t%s¥t%s¥t", $1, $2, $3)     # print fixed data
        z = split($4,k," ")
        if (k[1] >= 10) printf k[1]             # print total
        else printf " " k[1]
        for ( i = z-6+x; i <= z-1+x; ++i )       # shift seat/"欠" data
```

```
                printf "  " k[i] " "
        print "  " k[z+x] " " }
    print "  表示始め = " j[z-6+x] ", 表示終り = " j[z+x]
                } }

#8-3 prompt user for other shiftings
    { if ( z-6+x > 2 )
        { printf("  Backspasei(またはCtrl+H)キーで表示の範囲を変えられ")
        printf("ます(q to quit)。")
        }
    else { print "  表示範囲の変更を終了しました。"
        exit }
        }
```

## プログラムの出力例

①過渡的な単票型プログラム <class>, <check> によるもの

```
    ■      テ      ス      ト      版
    出席状況      6302教室        9月 4日  16:58      在室者 3人
    ------------------------------------------------------------------------
    9712XXX ××    ××   Xxxxx____Xxxx_____   xxxxxx7        欠
    9731XXX ××    ××   Xxxxxx___Xxxxxxxx__  xxxxxx71        欠
    0032XXX ××    ××   Xxx______Xxxxxx____  xxxx02          欠
    9731XXX ××    ××   Xxxxxx___Xxxxx_____  xxxxxx71        17  15:45
    9711XXX ××    ××   Xxxxxxxx_Xxxxx_____  xxxxxx7         20  14:41
    staff   ××    ××   Xxxxxxx__Xxxxx_____  xxxxxxx         01  logged
```

ふたつのプログラムの差異は，前者が <sheet> という一時ファイル，後者が <yymmddⅡ> などの個別ファイルに入って保存される点にだけあり，他に異なる点はまったくない。項目は，学籍番号・氏名（漢字）・ローマ字表記・ログイン名・座席番号または欠・ログオフ時刻または"logged"（"still logged in"の一部表示）になっている。

②本報告のデータベース型プログラム <table> によるもの

■ テ　ス　ト　事　例　6302教室　9月4日

| | | | 計 | 0926 | 0927 | 0928 | 0929 | 0930 | 1001 | 1002 |
|---|---|---|---|---|---|---|---|---|---|---|
| 9712XXX ×× | ×× | xxxxxx7 | 0 | 欠 | 欠 | 欠 | 欠 | 欠 | 欠 | 欠 |
| 9731XXX ×× | ×× | xxxxxx71 | 0 | 欠 | 欠 | 欠 | 欠 | 欠 | 欠 | 欠 |
| 0032XXX ×× | ×× | xxxx02 | 0 | 欠 | 欠 | 欠 | 欠 | 欠 | 欠 | 欠 |
| 9731XXX ×× | ×× | xxxxxx71 | 11 | 17 | 17 | 17 | 17 | 17 | 17 | 17 |
| 9711XXX ×× | ×× | xxxxxx7 | 11 | 20 | 20 | 20 | 20 | 20 | 20 | 20 |
| staff ×× | ×× | xxxxxxx | 11 | 01 | 01 | 01 | 01 | 01 | 01 | 01 |

表示始め = 0926, 表示終り = 1002

Backspace(またはCtrl+H)キーで表示の範囲を変えられます(q to quit)。

項目は，学籍番号・氏名（漢字）・ログイン名・出席時間数・（月日の下に）座席番号または「欠」が表示される。3名の教員が分担する総合コースの場合には，各教員の担当こま数が4～6なので，この書式で不便はないが，右の余白部が終わったあとの表示は，当然各2行目以降に「折り返し」表示されてしまい，読みにくくなる。そこで，「計」項目までの表示を固定し，その右側に最終7回分の出欠データを表示するようにプログラムを修正した。この表示範囲を，後退（バックスペース）キー（Ctrl+Hが同等キー）入力によって，疑似的にヨコ方向にスクロールさせている。

## プログラムの論理

以下簡易データベース型プログラムについて説明するが，①～⑧は，プログラムの中の処理群（rule）#1～#8に対応している。

①はじめに，プログラムを始動した時点での月日・時刻・曜日の各データを"date"命令でえているが，比較照合する"last"ファイルの構文を意識して，たとえば「09月04日」を「9月」「4日」に変えている。「0904」などはデータベース型項目itemとして照合用・記載用に利用する。教室の"date"データは日本語表記であたえられるが，その処理のためのsplit()関数の区分文字（セパレーター）には漢字をつかっている。また，プログラム使用時刻の変数nowは，後の授業開始時刻・履修者ログオフ時刻などとの比較のために，午前00時00分からの「分」の通算値にしてある。

②次にこの時間データをもとづいて，教室・クラス名簿・暫定ファイルを選択する。ここ

では，報告者がじっさいに担当している3授業が，曜日・時限で選択されるようになっているが，この部分を書きかえれば，他のコンピュータ授業担当者も，本試作プログラムを利用できる。4番目には，このプログラムのテスト時にサンプリングした実例にもとづく架空クラスのデータを記しており，これは，実際の3授業の出席点検以外の日・時刻にこのプログラムを使ったときに作動するようになっている（作動日は9月4日に固定してある）。

③"last"ファイルのなかから，授業当日の月日・使用教室（ラボ）の2条件に合致する者だけを選択するが，この部分は日付が1桁の場合と2桁の場合に分ける。"last"ファイルが空白（スペース）分割型の構造で，前者が9項目（フィールド）構文，後者が10項目構文になっているためである。だから，条件文は日付の桁でなく，項目数（NF）で書いても良い。該当者が"still logged in"か，授業開始時刻以後にログオフしている場合にかぎって，ログイン名を引数とする連想配列seatをつくり，それに座席コードを代入する。ログオフ時刻（lt）と授業開始時刻（bt）とを比較しているのは，同日・同教室で他授業履修・自習などしている場合に「出席」と誤認したり，さかのぼって古いほうの値を上書きしてしまったりするのを避けるためである。

④サンプルリストは，②で選択される3授業の時間帯外にこのプログラムを作動させたときに選択されてプログラムをテストするもので，名簿末の3名にだけ座席番号が入り，他の学生には「欠」が記載される。異なる日にテスト走行をくりかえすと，この3名についてだけ出席数が加算されて行く。しかし，"last"ファイルが更新されて，「9月4日」のデータが失われてしまうと，テストリストとしての意味を持たなくなるので，その場合に，"last"ファイルに変更があったことを警告し，サンプルリストの変更または削除をうながしてプログラムを停止するようにしている。

⑤授業中，何度でも「出席取りなおし」ができるようにした部分で，この部分から項目読みこみ区分（FS）を作表（タブ）文字に変える。ここでは，仮作製ファイルの2行目末尾の項目（「計」または「0925」など）と当日のitemとが異なる場合にだけ，仮ファイルを本ファイルにコピーする。これで初回分も含め，同じ授業での「取りなおし」をすることが可能になる。つまり，実際には次の週の出欠点検のときになって初めて，本ファイルの書きなおしが行われることによって同じ日の「取りなおし」を可能にしているので，最終授業までのデータは仮ファイルの方にだけ残っていることになる。

⑥履修者名簿の学生ひとりひとりについて，"last"ファイルから送られてきた座席番号の連想配列がつくられているかどうか（出席しているかどうか）を点検し，存在する場合には，本ファイルの記事の右側に座席コードを記入して「計」の値を増やす。「計」が2桁になったときは，左側の空白を消し，表示がずれないようにする。連想配列がない場合には，「欠」の文字を同じ場所に記入する。これらの記入作業は，すべて仮ファイル側に対して行い，本ファ

イルへのデータ記入は行わない。出席の「取りなおし」を行ったときには，"last" ファイルの点検以降の作業を前の週の本ファイル記録との比較でくりかえすだけにしてあるので，授業時間中なら，何度でも出席の「取りなおし」ができる。

⑦最後に，データ記入が終わった仮ファイルを画面に表示する。この画面は学生卓のモニター側でも確認できる。表示は余白の幅に制限され，7週分までしか出力できない。それ以上表示しようとすると，次行に「折りかえし」されて読みにくくなるので，8週分以上の出欠データがあるときには，最終7週分のデータだけを 表示するようにした（もちろん，仮ファイル・本ファイルとも，ファイル本体には全データがある)。この 出欠データを編集ソフト「秀丸」を介して印刷すれば，授業用の資料にできるが，印刷する場合，プロポーショナル書体だと表が読みにくくなる。また，表の転送にはコピー＆ペースト方式のほうが良い。仮ファイル名を教室に設定してある「M ドライブ」から「秀丸」へのドラッグ＆ドロップ方式で出力すると，「文字化け」が生じてしまうからである。学期末や年度末に全データを印刷したい場合にだけ，「文字化け」処理を行った後に，後者の方法で横長の用紙に小ポイント文字で出力する。そのときにだけ，仮ファイル "intro"（入門)・"sogo"（総合科目)・"hist"（歴史ゼミ)・"list"（テスト用）などの名前を意識する必要が生ずる。説明⑥のように，最終データが仮ファイル側にのこっているためである。

⑧ふつうは最終7週分の表示範囲で十分だろうが，これを変更したい場合もありうるので，キーボードからの標準入力によって，表示範囲を変更できるようにした疑似的なヨコ方向スクロール部分である。表示月日の範囲が出力されたあとに，入力要求（プロンプト）が示されるので，後退（Backspace または同等の Ctrl+H）キーを押すと，画面が 消去されたのちに1週間分まえに変更された範囲が表示される。画面を毎回消去するのは，画面のタテ方向スクロールの「うるささ」を感じさせないためである。目的にあった画面を印刷するのには，⑦と同様に，「秀丸」への コピー＆ペースト方式を使う。"q"，"Q" あるいは "quit" で，この疑似ヨコ方向スクロール画面は停止し，プログラムが終了する。

## データファイルの仕様

①タイトル行：科目名，使用教室，曜日，時限，その他必要な事項を1行（以上）記入する。空白は1バイトでも2バイトでも良いが，作表（タブ）文字は使用できない。タイトル行は省略しないほうが良いが，省略してもエラーにはならないと思う。

②項目行：作表（タブ）文字を5回入力して左方の空白部をつくってから，「計」などの2バイト文字1字（または1バイト文字2字）を入力しておく。この項を2バイトにしておかないと，そのあとに続く項目部分の表示がずれてしまう。

③履修者名簿：学籍番号（7バイト）・氏名（漢字6字＝12バイト，中間の空白には2バイトの空白を使ってあるが1バイトの空白でも良い）・ログイン名（8バイト，それより短いものには末尾に1バイトの空白を必要個数加える）・″ 0″（左側の空白は，出席数の桁上がり用）の4項目であるが，項目と項目のあいだの区分（file separator）には，かならず作表（タブ）文字を使う。そうしないと，あとの表示が乱れてくる。

タイトル行部分の空白にはTab（作表）キー を使わず，全角スペースを使う

```
■「歴史ゼミナール」          6406教室  水曜日    Ⅳ限
←     ここの 空白は Tab キー5回      →  計
9721XXX □□    □□    xxxxxx72        0
9741XXX □□    □□    xxxx7           0
9741XXX □□    □□    xxxxxx7         0
9811XXX □□    □□    xxxxxx8         0
9811XXX □□  □□□    xxxxx8          0
9812XXX □□  □□□    xxxxxx8         0
9812XXX □□    □□    xxxxxx8         0
```

Tab　全角空白　Tab　末尾空白 Tab　空白1バイト

④ファイル：本ファイルと仮ファイルのふたつが必要だが，ファイル名は自由で，なにも制限はない。プログラムの #2 部分を変更し，一方のファイル名から，他方のファイル名が自動的に生成されるようにしたほうがスマートだったかも知れない。

## 謝　　辞

本プログラムの試作については，経済科学部の坂口通則氏，廣光清次郎氏，情報センターの山崎昭弘氏，記谷康之氏から，とくに有益な示唆，助言，教示，協力をえた。これらの人たちの支援がなければ，情報科学を専門としていない報告者が今回のプログラミングを試み，修正の末，完了することなど到底ありえなかった。ここに記して感謝する。

## Summary

The log-in record of the direct users of UNIX OS are stored in the system and can be referred by the "last" command. The author has tried to link the data with the class student lists by using the 'awk' language to check the attendance and the seats of the students in the classrooms. The 'awk' is a pattern scanning and processing language, collabaroted by Aho, Wein-berger and Kernighan till 1985. After making relatively primitive programs, "class" and "check" for mere one week result report, he could achieve at last to write a more useful data-base-like program "table" for the last seven week data, indicating the seat-numbers or "欠" (absent symbol in Japanese) of each student. The given range of student attendance can be reviewed retroactively by pushing backspace or control+H key. This program can be utilized naturally by other staff, only by revising #2 part of it. If this simple effort by an amateur staff could stimulate the students in the "IT-Age", it might give him a great pleasure.