THE STREETWALKER FILE

Harold P. Sieglaff
Computer Techniques Analyst
City of Phoenix
Arizona 85003

A STREETWALKER FILE is one ordered by the way a person walks down a street. The particular file under discussion is for a water meter reader walking down a street reading meters, hence the name STREETWALKER FILE.

The file organization is that of a PAGE LINE. A PAGE LINE file is one in which the location of a record is determined by the values of PAGE and LINE. It may be considered a 2 dimensional array.

The file may be considered a giant deck of cards (1 card per line). The file may be processed by

(1) ADDing new cards (records lines)
(2) DELETing (discarding) old cards (records, lines)
(3) MOVing old cards (records, lines) from one place to another.

A PAGE LINE file may be used for source pgm maintenance, text editing, or to describe a "random walk." The PAGE LINE file described here is used in the new water billing system of the CITY OF PHOE IX.

PGM 1 Initialize STREETWALKER FILE
Put secondary pages in free space chain

2 Create ADD/DELETE records from

ADD AFTER KILL MOVE ADD AFTER input records

3 Sort the output of PGM 2 by PAGE LINE between the LINES values

4 Check PGM 3 output for

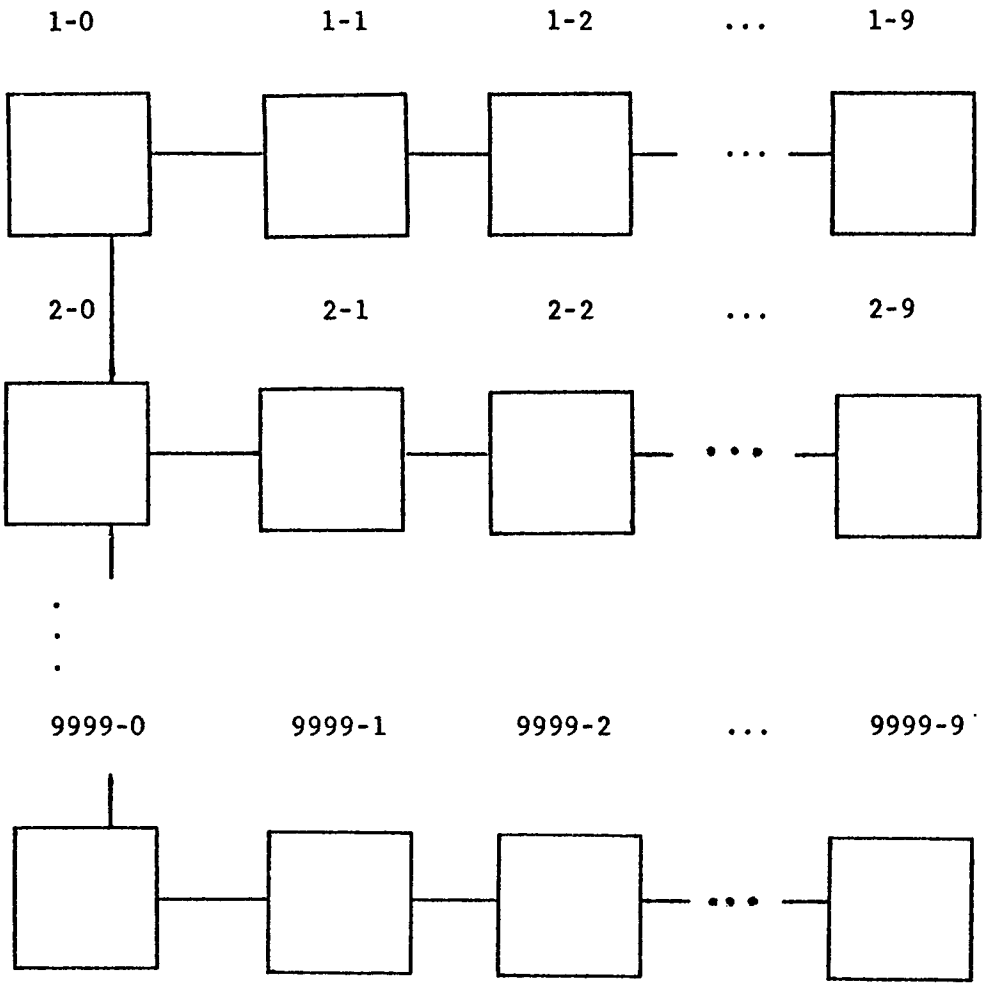
(a) duplicate records (you may not KILL AND/OR MOVE the same record)

(b) a page with more than X lines

5 Merge the STREETWALKER FILE and the output of Pgm 4

To ADD DELETE MOVE records run PGMs 2-3-4-5.

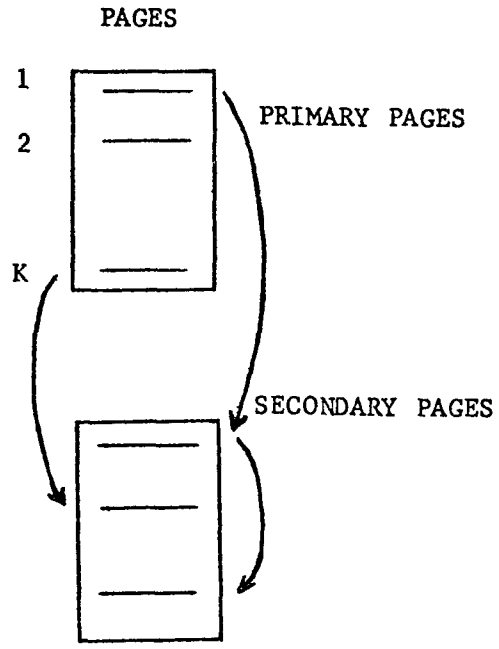
The file may be pictured as follows:



Each page contains:

1) date last written
2) # of pages comprising this page (PRIMARY + # of SECONDARY PAGES)
3) # of records on this page (# of records on the primary page + # of records on the secondary pages)
4) # of records on this single page
5) # of the next secondary page (if any)
6) the lines (records) of the page

Each page is a chain of pages consisting of 1 primary page & 0-9 secondary pages. Each page of a chain is numbered ascendingly as follows: 1-0 1-1 1-2 ... 1-9
PAGE NNNN0 = primary page NNNN1 - NNNN9 = secondary pages



In the example PAGE 1 has 3 links (pages)
PAGE 2 " 1 link (page)
PAGE K " 2 links (pages)

Card input format

ACTION	SEQ # START	SEQ # END	ACCT #	ADD AFTER SEQ #	AS THE Kth RECORD*
ADD	---	---	YES	YES	YES
MOVE	YES	YES	---	YES	YES
KILL	YES	YES	---	---	---

*K is a "between the lines" value. It simulates assigning a value = the order in which the cards are input

K need not be used if the input is in this order:

1) KILL actions in order by SEQ # START
2) ADD and MOVE actions in order:
   a) by ADD AFTER seq # values
   b) in which they are to be ADDed AFTER seq # if the same ADD AFTER seq # appears on 2 or more actions

KILL = DELETE (1 action)

MOVE = DELETE and ADD (2 actions)

ADD = ADD (1 action)

7 program street suite

WAB170	PGM	1	Initialize RANDOM ST FILE
172		2	Sort new streets
173		3	Add new streets to RANDOM ST FILE → SEQ ST FILE
176		4	Copy SEQ ST FILE to RANDOM ST FILE & set alpha pointers
179		5	Print RANDOM ST FILE
185		6	Correct st spellings by time value (RANDOM ST FILE) copy RANDOM ST FILE to SEQ ST FILE
186		7	Sort SEQ ST FILE by st name & set chain head pointers
176		4	Copy SEQ ST FILE to RANDOM ST FILE & set alpha pointers
179		5	Print RANDOM ST FILE

To add new streets RUN PGMS 2-3-4-5

To correct street spellings RUN PGMS 6-7-4-5

RECOVERY FROM DISASTER

Let # streets = N + K

If RANDOM ST FILE is destroyed recover as follows

RUN 1 PGM 1 Initialize RANDOM ST FILE

RUN 2 PGMS 2-3-4-5

Add 1st N streets for which J alpha position = J time value J = 1 - N

RUN 3 PGMS 2-3-4-5 Add K streets named Z1 ... ZK

RUN 4 PGMS 6-7-4-5

change spelling of streets Z1 ... ZK
to the correct spelling of streets N + 1 ... N + K

Machine Independent Sequential Files

The following technique allows for machine independent (e.g. tape) files. Records are constructed so that each 6 or 8 bits (character size) is a human readable, stand alone, printable symbol.

Data fields consist of 1 or more characters (symbols) in a string. For example,

```
DATA  =  a  4  character  alphanumeric  string
99953 =  a  5  character       numeric  string.
```

External to the file on machine independent typewriter paper is a data description table or data definition table DDT of N entries where N = # of fields in the record. Each table entry consists of 3 parts:

(1) type of field

- X = alphanumeric
- 9 = numeric unsigned
- S = numeric signed
- $ = numeric dollar and cents
- D = numeric date with implicit/symbols between each 2 characters

(2) # of characters in the field

(3) # of decimals in the field (usually 0 or 2 in COBOL)

In general, there are only 2 types of fields alphanumeric and numeric. If a table occurs in a record its individual entries can be identified separately in the data description table DDT.

Of course, the DDT can be created in the core of the machine in which the record is to be processed. To convert from machine 1 to machine 2 one need only convert characters one by one.

This kind of record makes debugging easier in that every 6 or 8 bits is a printable character unlike the unprintable characters comprising fixed or floating point binary or packed decimal numbers. To speed processing the fields of the record may be moved in core to fields described as binary or packed decimal.

The following table illustrates the technique:

```
RECORD DESCRIPTION  X     = ALPHANUMERIC
                    NON X = NUMERIC  SIGNED OR UNSIGNED

  *  10   15   20   25   30   35
  *
  *  #    TYPE SIZE DEC  SUM  ITEM NAME
  *
  *    1     9 11   0     11  ACCT #
  *
  *    2     X 24   0     35  NAME
  *
  *    3     S  9   2     44  BALANCE  PIC S9(7)V99
  *
  *    4     9  9   0     53  SOCIAL SECURITY #
  *
  *    5     D  6   0     59  DATE ARRESTED  MO DA YR
  *
  *    6     9  4   0     63  TIME ARRESTED  HR MIN
  *
  *    7     X  1   0     64  ACTIVE OR INACTIVE RECORD
  *
  *    8     9  4   2     68  DEPOSIT AMT
  *
  *    9     9  7   2     75  CURRENT BILL
  *
  *   10     9  5   0     80  CURRENT CONSUMPTION
  *
  *   11     X 42   0    122  COMMENT 1
  *
  *   12     X 42   0    164  COMMENT 2
  *
  *   13     S  6   0    170  FINAL READING < 0  = NO READING
```