Roach

To:	Distribution
From:	T. H. Van Vleck
Date:	01/13/75
Subject:	Indexing Multics Manuals

This memorandum describes a new method for generating indices for Multics manuals.

Old Method

The indices to the FORTRAN and BASIC manuals were generated by a set of programs which ran on Multics. Given a file listing words and phrases to be indexed, the programs called runoff to produce a temporary copy of the final runout in order to get the correct page numbers, and scanned the runout for all the terms to be indexed. All "hits" were recorded in a data structure with threads running forward, back, up, and down. A second program sorted this list structure and produced a runoff file for the index. The entry for one term might look like this:

```
   abbreviations
     3-1ff, 3-8, 3-101
```

this entry records the page numbers on which any of the terms (say) "abbreviation," "abbrev," or "profile" occurred. Now, indexing on "profile" is unfortunate, since the section for the profile command will cause "false hits." And yet, if we don't index on "profile," some pages which really ought to be pointed to by the entry for "abbreviations" might slip by.

For the language manuals, there were few enough false hits that they could be removed by manual editing. But for the MPM commands section, the false hits far outnumber the genuine (consider "do" and "where"), and manual intervention is required to separate references to words used in more than one sense (e.g. "link"). The magnitude of the manual editing task for the MPM is unacceptably large, especially since the MPM will be subject to much more re-issue and revision than the language manuals.

New Method

Instead of making an index which contains only topic and page reference, the new method will produce an index which should also be more useful to the person trying to find out about the system. An entry in the new index will look like this:

```
command language
  execution
    abbrev 3-3
    answer 3-12
    enter_abs_request 3-126
    exec_com 3-130
    memo 3-209
    walk_subtree 3-335
  expansion
    abbrev 3-3
    do 3-100
    get_com_line 3-166
    set_com_line 3-313
  see active functions
  see directory
  see search rules
```

The major heading will be followed by any number of subheadings. Under each subheading, both the command name and the page number will be shown.

Implementation

Instead of marking the places where a hit shouldn't happen, we will modify the runoff source of the MPM and other documents to show where a hit should occur. A line of the form

```
.if hit "command language~execution"
```

will be inserted wherever a hit is desired, for example at the beginning of the writeup for abbrev. The macro file hit.runoff will be controlled by a global variable which tells whether indexing is on or off. It might look like this:

```
.ts %indexing%
.ex index_hit %Parameter% %textloc% %Section% %Page%
```

That is, if indexing is on, it calls a simple PL/I program to add a line to an ASCII file.

The ASCII file generated by collecting all the lines generated by the index_hit program will look something like this:

```
command language~execution~abbrev~3-3
command language~execution~answer~3-12
command language~execution~enter_abs_request~3-126
command language~execution~exec_com~3-130
command language~execution~memo~3-209
command language~execution~walk_subtree~3-335
```

```
        command language~expansion~abbrev~3-3
        command language~expansion~do~3-100
        command language~expansion~get_com_line~3-166
        command language~expansion~set_com_line~3-313
        command language~see active functions
        command language~see directory
        command language~see search rules
```

From one to four fields may occur in each line, separated by the tilde character. This file will be sorted and then processed by a simple PL/I program to generate a single column of running text.

Although it is possible to write a program to create a two-column format index, the current plan is to create the final output pages by manual cutting and pasting.

Automatic Generation of Table of Contents

Currently, the tables of contents of Multics manuals are generated by hand. After a final runout copy has been produced, it is examined for the correct page numbers for each level of heading. Slight modifications to the runoff source of the documents will allow us to generate the table of contents automatically as well as assist in indexing.

Honeywell's standard documentation format defines first, second, third, etc. level headings, specifies the type font for the heading, and requires that the first three levels of heading appear in the table of contents. For example, first level headings should be all capital letters, underlined. In order to collect the locations of the headings into a segment so that the table of contents can be generated automatically, we will replace the raw text of the heading with a call to a runoff macro as follows:

```
        .if first_level_head "Constructing and Interpreting Names"
```

When this macro is called, it will emit the heading in the appropriate format (underlining and translating to upper case if necessary), and if the indexing switch is on will also add a line to another ASCII file which can be used to generate the table of contents later. The heading macros will also set the runoff variable "textloc" to the title so that the index_hit macro can use this variable. (For the MPM command and subroutines volumes, textloc will be set to the command or subroutine name.)

Changes to be Made

Modifications will have to be made to the runoff source of the various documents to be subjected to this new scheme. Most of these changes can be performed automatically by PL/I or qedx programs.

New typing conventions must be documented for future manuals.

The indexing and table-of-contents generation procedures must also be documented, and the programs which are used documented and submitted to the tools library.