..\INVERT3\INV3MAN.TXT

USERS GUIDE TO INVERT3

0. INTRODUCTION TO THE PROBLEM

A finderlist can be a much more valuable resource than an index if appropriate attention has been paid to its construction. Byron Bender recognized this in compiling the Marshallese dictionary (Abo et al., 1976), in which he notes (p. xxxiv):

> "The [finder]list was also compiled with another type of user in mind, the student of Marshallese culture interested in seeing the total set of vocabulary associated with important topics of life in the islands. Such readers can find pulled together in one place many of the words associated with subjects such as fish and fishing methods; breadfruit, pandanus and coconut culture; other plant names; games; winds, tides, sailing terms, canoe parts; foods, smells, and so forth. These groupings present an avenue to the culture of the islands through the words of the language."

The same point was made by Jim Kari, editor-in-chief of the Koyukon dictionary (Jette' and Jones 2000, p. 839):

> "The index offers many insights into the ways in which the Koyukon classify and view their world, and areas of special study can be pursued using this index as a guide. There are many long entries with numerous words. For example, types of snow and verbs of snowing are given at SNOW. Other extended entries include MEAT, ICE, and WATER."

In the finderlist of the Alabama dictionary (Sylestine et al., 1993) Tim Montler also made a point of including some domains of cultural interest as keywords (p. xxvii):

> "Semantic class entries are very large entries in which words relating to a particularly important semantic class (for example, plants, numbers, clothing, and kin terms) are listed. Some of the semantic class labels are: agriculture, amphibians, birds, body parts, ... , tools, trees. The semantic class entries are useful for finding words that one is unable to locate under a certain English word, as well as for immediate information about items that are important in Alabama culture."

The agenda presented by these authors can be pursued much further, resulting in even more comprehensive finderlists, by recruiting keywords (starred words placed in definition bands, in the markup system for generating finderlists) from the total range of topics of cultural and linguistic interest beyond the mostly physical subjects mentioned so far. Some of them will be mentioned briefly later. But the bulk of this document is about minor extensions to the current finderlist program that

address a difficulty with using large entries generated by the program, a problem which some authors have tried to remedy on an ad-hoc basis. A few other improvements to the old finderlist program are also introduced.

The current program can produce finderlist entries containing many long entries with dozens or hundreds of items in each, listed in alphabetical order. Due to their length and semantically arbitrary order, these lists can be inconvenient to work with. In the Marshalese dictionary, for example, there are 59 items under "sail" (from "a canoe with its sail flapping" to "windward overlay of sail"), 83 items under "pandanus", 131 under "plant" and nearly 300 under "fish"--and more than 300 in the updated on-line version (www.trussel2.com/mod). The Klallam dictionary (Montler 2012) lists 70 relationship terms under "family" (from "adult younger sibling or child of parent's younger sibling" to "youngest child in the family"), 50 items under "canoe", 75 under "break", and and 205 placenames under "placenames".

The lists would be more useful if each were organized into logical groups under subkeywords (and even sub-subkeywords). Kari actually took the trouble to do this manually for some finderlist entries (loc. cit.):

> "The more elaborate entries are broken into subsections such as "types of" and "parts of" or "products of". (For example see the entries for FISH, SALMON, MOOSE, CARIBOU, BOOT, HOUSE, and PEOPLE.)"

Thus the 83 items under FISH are separated into "Types of fish", "Parts, anatomy of fish", and "Fish products", each group containing a more manageable 20-30 alphabetized items. The 85 items under PEOPLE are subgrouped under "General terms", "Koyukon regional band names and named peoples", "Other Athabaskan peoples", "Other named ethnic groups", and "Legendary peoples". Montler appears to have done similar subgrouping for some of the larger finderlist entries in the Alabama dictionary, although without explicit subkeyword labeling or separation in the format: the body parts entry, for example, is subgrouped approximately into "human body parts", "animal body parts", and "bodily secretions". Similarly, in his more recent Klallam dictionary (Montler, 2012), the items under "break" are clumped into these semantic groups: "break something external to the body", "to be broken", "break some body part".

The trouble with doing subgrouping manually AFTER the program has been run is not only the amount of error-prone labor required but also the fact that all that labor is lost when the program is run again. What is needed is a convenient method for the user to indicate, in the keyword markup in the original definition bands, the desired subgrouping BEFORE the program is run.

1. SUBKEYWORDS

An extension of the "invisible" keyword markup convention to do this is introduced with this version of the finderlist program, INVERT3.

The basic device is to include a sequence of more than one keyword in the "invisible" keyword bracket (angle brackets). The second (and third, etc.) keyword is taken by INVERT3 to be the the sub- (and sub-sub etc.) keyword(s). (The old program ignores any additional words after the first in angle brackets.) The following short entry would generate the finderlist entry shown below it:

```
.hw  kwarkor
df  <*pandanus parts_of> pandanus leaf used for rolling cigarettes

.kw  pandanus
..kw  parts of
ph   pandanus leaf used for rolling cigarettes:: kwarkor
```

The generated subkeyword band, marked by two dots, follows the same convention that is used for subheadwords (e.g., ..hw) in the dictionary itself. Subsubkeywords would have three dots, etc.

Keywords in the bracket are terminated and extended in the same manner as they would be outside of brackets. A more fleshed-out finderlist entry, generated from several dictionary entries, might look like this:

```
.kw  pandanus
..kw  parts of
ph   dry key of pandanus fruit:: pej
ph   leaves near pandanus stem:: ainmwak
ph   pandanus leaf used for rolling cigarettes::  kwarkor
..kw  uses of
ph   a game, pelting one another with lighted pandanus keys or coconut
  husks:: buwaddel
ph   pandanus mat for sail cover:: atro
```

Note that to be part of a subkeyword sequence a word must be in the angle brackets. This may require duplicating a word, as "pandanus" in the above df band, whereas if no subkeyword were desired then the entire bracket in this case would be unnecessary, with the * being attached to the word "pandanus" in the definition itself. As usual, the pair of angle brackets and their contents do not appear in the ph band, the "phrase", in the finderlist entry. Any words preceding the first starred word in angle brackets are ignored.

Any number of invisible brackets may be used within a definition. This entry

```
.hw  buwaddel
df   <*pandanus uses_of> <*games> a game, pelting one another with
```

```
     lighted pandanus keys

   .hw  del
   df  <*pandanus products_of> <*foods> pandanus pudding cooked in hot rocks
```

would generate

```
   .kw  foods
   ph   pandanus pudding cooked in hot rocks:: del

   .kw  games
   ph   a game, pelting one another with lighted pandanus keys:: buwaddel

   .kw  pandanus
   ..kw  products of
   ph   pandanus pudding cooked in hot rocks:: del
   ..kw  uses of
   ph   a game, pelting one another with lighted pandanus keys:: buwaddel
```

Angle brackets are not needed, and are not dealt with, within angle brackets, since everything with angle brackets is already "invisible". Keyword truncation, by default with the | character, is also not recognized.

2. CROSS-REFERENCES BETWEEN KEYWORDS

If a subkeyword in angle brackets is starred, then a cross-referece to the main keyword in the brackets is generated along with the cascade of intervening subkewords:

```
   .hw   pako
   df  <*fish kinds_of *shark> general term for shark
```

would generate these entries, with the new band, xr, containing the cross-reference:

```
   .kw  fish
   ..kw  kinds of
   ...kw   shark
   ph   general term for shark: pako

   .kw   shark
   xr   fish, kinds of
```

No cross-reference is made if the subkeyword is not starred. A starred subkeyword is not made into a main keyword like the first starred word in the brackets is; it only generates a cross-reference. But if you want it also to be a main keyword, you can star the word in the definition or in a separate pair of angle brackets, e.g.,

```
.hw   pako
df  <*fish kinds_of *shark> general term for *shark
```

generating:

```
.kw  fish
..kw  kinds of
...kw   shark
ph   general term for shark: pako

.kw   shark
xr   fish, kinds of
ph   general term for shark: pako
```

3. ABSOLUTE CROSS-REFERENCES

An "absolute cross-reference" is one that is not generated from a subkeyword, but inserted explicitly without reference to any dictionary entry. This can be done easily using the subkeyword conventions, by making an "empty entry" in the dictionary, one with a blank headword and with definition bands containing only invisible brackets, e.g.

```
.hw
df   <*profanities  *curses>
df   <*swear_words  *curses>
df   <*family   *relatives>
df   <*family  *kinship>
df   <*dance  *song>
df   <*life-cycle marriage *wedding>
```

leading to these cross-reference entries in the finderlist:

```
.kw   curses
xr   profanities
xr   swear words

.kw   song
xr   dance

.kw   kinship
xr   family

.kw   relatives
xr   family

.kw   wedding
```

```
    xr   life-cycle, marriage
```

An extensive system of cross-references can help the reader who is looking for some concept but does not know exactly what keyword to look up in the finderlist.

4. "INVISIBLE" PARTS OF KEYWORDS

Not only an entire keyword but also parts of a keyword may be rendered "invisible" by placing them in angle brackets. This may be useful for disambiguating homonymous keywords when taken out of the context of a phrase, or otherwise when a slightly different form of the word would be more appropriate when heading an entry in the finderlist than when embedded in the phrase. Thus the dictionary entries

```
    .hw   at
    df    *bow<_(of_boat)> waves from a ship

    .hw   lippoanw
    df    shoot a *bow<_(weapon)> and arrow; *bow<_(weapon)>

    .hw   lwoboarwa
    df    *bow<_(of_boat)>|sprit

    .hw   mwajid
    df    to *bow<_(v.)>
```

would generate separate finderlist entries for the three meanings of "bow":

```
    .kw   bow (of boat)
    ph   bowsprit::  lwoboarwa
    ph   bow waves from a ship::  at

    .kw   bow (v.)
    ph   to bow::  mwajid

    .kw   bow (weapon)::  lippoanw
    ph   shoot a bow and arrow::  lippoanw
```

The old method of using subscripts ($1, $2, etc.) on keywords is still available since the program leaves them alone as part of the keywords. In fact, they can be used in conjunction with the above device, to control the sequencing of the homonyms:

```
          *bow$1<_(of boat)>
          *bow$2<_(weapon)>
          *bow$3<_(v.)>
```

Their deletion, if desired, can be postponed until formatting of the finderlist for publication.

5. SEMI-INVISIBLE BRACKET

After all the invisible brackets have been removed, if there remains a closing angle bracket in the definition, then everything up to that bracket will also be removed from the phrase in the generated finderlist but unlike paired brackets can be a signal to the program that formats the dictionary to NOT delete the material from the dictionary. This "semi-invisible" bracket convention can be useful when some initial portion of a phrase seem necessary in the definition but would seem redundant in the finderlist because of a preceding keyword or subkeyword.

6. INVISBLE BRACKETS FOR COMMENTS

At no additional programming cost, invisible brackets can be used to enclose comments anywhere in the definition (but not within another invisible bracket).

7. EXPLICIT SEQUENCING OF SUBKEYWORDS AND PHRASES

Another problem with finderlists that authors have had to deal with manually is the sequencing of phrases in an entry when some sequence other than alphabetical order is desired, as with days of the week, months of the year, or taxonomic order. This problem can now also occur with sequences of subkeywords when a particular order, such as "types of", "products of", "uses of", is desired. Under the keyword "birds", subkeywords in a particular order may be desired, e.g., "seabirds", "wading birds", "ducks and geese", "raptors", etc.

In the Koyukon finderlist for instance, under "month", a manually inserted subkeyword "Months according to traditional lunar calender, in order:" heads a list that had to be manually rearranged into numerical order from the alphabetical order produced by the program:

```
first month in the traditional lunar calendar, around December
second month in ...
third month in ...
fourth month in ...
fifth month in ...
```

In the Shinzwani finderlist also, manual post-processing was required to re-order the days of the week (under the keyword "week") from the alphabetical order provided by the program.

A limited facility is offered in INVERT3 to specify the sequencing of particular phrases and subkeywords. Lists of individual words and phrases, such as "Saturday", "Sunday", "first month", "second month", have to be placed, in the desired sequence, in the SPECS file in a particular format (simple Spitbol statements). They are called "incipits", since they specify as much of the BEGINNINGS as necessary to uniquely identify the target phrases and subkeywords.

The specified sequence of incipits applies "globally", i.e., to all phrases and subkeywords identified by the incipits. Hence the incipits need to be unique so as not to apply to unintended targets. The same list of incipits applies to both subkeywords and phrases, but of course subkeywords and phrases would never get intermixed, nor would subkeywords of different levels. There is no way to target only subkeywords or only phrases. Incipits are case-sensitive: "polish" is diferent from "Polish". In a given entry any subkeywords, and in an entry or subentry any phrases, that are not identified by incipits follow those that are, in alphabetical order. See the first entry for "week" in the example below.

The format of the SPECS statements is as follows: the list must be preceded by the statement

```
DEFINE('SEQ(I)')
```

(because, due to the position of the SPECS file near the beginning of the program, the SEQ function has not yet been defined, though the code for the function is present later). This is followed by a list of calls to SEQ with the incipits (I) that, simply by being in the given order, specify the desired sequencing, e.g.

```
SEQ('Saturday')
SEQ('Sunday')
SEQ('Monday')
    etc.
```

Different lists of incipits may be run-on one after the other, or they may be separated by a call with a null argument SEQ(), as in

```
  .
  .
  .
SEQ('Thursday')
SEQ('Friday')
SEQ()
SEQ('seabirds')
SEQ('wading birds')
SEQ('ducks and geese')
  .
```

```
          .
          .
```

(As are all SPECS statements, these are SPITBOL statements, and must conform to the syntax of that language. In particular, note that they do not start at the left margin ["column 1"].)

Each such break starts a new sequence counting from 1, so that if ever a list of phrases contained both birds and days of the week, say, they would get intermingled. The only reason to make separate lists is that there is a practical limit of 250 in each list. If there are duplicate incipits among the lists--not just in any single list--the latest-assigned (not necessarily the highest-valued) sequential position supersedes.

The following example shows how this works, assuming the above SEQ statements are in the SPECS file when the program is run. The following entries, simplified for illustration purposes,

```
      .hw   -ili
      df   two
      ..hw   mfumovili
      df    *Sunday (i.e., the second day of the *week)

      .hw   montsi
      df    one
      ..hw  mfumontsi
      df    *Saturday (i.e., the first day of the *week)

      .hw   -raru
      df   three
      ..hw   mfumoraru
      df    *Monday (i.e., the third day of the *week)

      .hw   -vira
      df   pass into; enter
      ..hw   mfumo wavira
      df   last *week
```

would produce these finderlist entries:

```
      .kw   Monday
      ph   Monday (i.e., the third day of the week)::  mfumoraru < -raru

      .kw   Saturday
      ph   Saturday (i.e., the first day of the week)::  mfumontsi < montsi

      .kw   Sunday
      ph   Sunday (i.e., the second day of the week)::  mfumovili < -ili
```

```
.kw   week
ph   Saturday (i.e., the first day of the week)::  mfumontsi < montsi
ph   Sunday (i.e., the second day of the week)::  mfumovili < -ili
ph   Monday (i.e., the third day of the week)::  mfumoraru < -raru
ph   last week::  mfumo wavira < -vira
```

whereas the same entries marked-up as follows:

```
.hw   -ili
df   two
..hw   mfumovili
df    <*week *Sunday> *Sunday (i.e., the second day of the week)

.hw   montsi
df    one
..hw  mfumontsi
df    <*week *Saturday> *Saturday (i.e., the first day of the week)

.hw   -raru
df   three
..hw   mfumoraru
df    <*week *Monday> *Monday (i.e., the third day of the week)

.hw   -vira
df   pass into; enter
..hw   mfumo wavira
df   last *week
```

would produce these entries:

```
.kw   Monday
xr    week
ph   Monday (i.e., the third day of the week)::  mfumoraru < -raru

.kw   Saturday
xr    week
ph   Saturday (i.e., the first day of the week)::  mfumontsi < montsi

.kw   Sunday
xr    week
ph   Sunday (i.e., the second day of the week)::  mfumovili < -ili

.kw   week
ph   last week::  mfumo wavira < -vira
..kw   Saturday
ph   Saturday (i.e., the first day of the week)::  mfumontsi < montsi
..kw   Sunday
ph   Sunday (i.e., the second day of the week)::  mfumovili < -ili
..kw   Monday
```

```
      ph   Monday (i.e., the third day of the week)::  mfumoraru < -raru
```

In the first version the days of the week come out as phrases in the main entry for "week"; in the second, they come out as subkeywords under the main entry. In either case, whether phrases or subkeywords, their proper sequencing is specified by the same sequence of calls to SEQ in the SPECS file. (Main keywords, of course, remain in alphabetical order.)

Also, in the first version, the phrase "last week" comes AFTER the list of sequence-specified phrases "Saturday" etc.; in the second, "last week", being part of the main entry for "week", comes BEFORE the subentries "Saturday" etc. begin.

Note also that the second version provides cross-references; the first does not. There are other possible ways to mark up the keywords in these entries, giving still other finderlist entry structures. These examples are meant only to show the effect of specifying sequencing.

8. PROXIES

A system of abbreviations, called "proxies", is offered to reduce repetitive typing of frequently-occurring long strings in the definitions, such as "products of" or "kinds of". Any string whatsoever may be represented by a proxy, and any string may be declared as a proxy, although to reap any benefit from using them the proxies should be significantly shorter than the strings they represent and these strings should occur with some frquency. Another consideration is that proxies should not be strings that normally would occur anywhere in the text. An easy way to ensure this is to start each proxy with some rare character that would never occur at the beginning of any normal word, such as "@". Thus you might assign "@prod" to represent "products_of", or "@f" to represent "fish", or even "@fty" to represent "*fish types_of".

This not a full-fledged macro system because the replacement is not recursive: a replaced string or any part of it is not subject to further replacement. Once a replacement has taken place, the cursor moves on, so to speak. Also, a longer proxy always applies before a shorter one that begins in the same way: if "@f" represents "fish" and "@fo" represents "fowl", a string in the text that begins with "@fo" will not be replaced by the shorter proxy to become "fisho". The longer one takes precedence.

Proxies are declared by statements you put in the SPECS file, and apply globally, i.e., to all definitions bands. They are applied to the band before anything else is done to it by the INVERT3 program. They can be declared in any order; the order does not matter. Declaring and applying proxies uses the transliteration table facilities already available in the Lexware core. Each declaration is of the form

```
      proxy  #  value
```

and the list must be preceded by a call to set up the PROXY table, e.g.,

```
          OPENTABLE('PROXY')
       '@prod'  #  'products_of'
       '@ko'    #  'kinds_of'
       '@f'     #  'fish'
```

(The # is defined in the INIT module. Again, these are Spitbol statements. Note the space on either side of the # sign.) When the proxies in this example are applied to the band

```
df   <*@f @ko *eel> a large black eel
```

it would become

```
df   <*fish kinds_of *eel> a large black eel
```

The program always applies the proxies before doing anything else with the band.

9. DEFINITION-EXTENSION BANDS

Often a definition can be very lengthy, perhaps including a lot of description or background. It is probably not necessary or desireable to copy all of that into the finderlist entry; only a succinct definition is needed, but the reader of the finderlist should be alerted to the fact that more information is available in the entry in the dictionary itself. This can be signaled with a discrete symbol placed next to the headword in the finderlist entry. In fact, a very similar device is deployed in the Koyukon dictionary's finderlist, again by maual postprocessing.

INVERT3 provides a way to do this. The succinct definition, to be used in the finderlist, is given in the regular definition band while the lengthy elaboration is relegated to an immediately following band, which by default has the same name as the definition band but with "x" appended to it, e.g., dfx, as in the following entry:

```
.hw  wadaha
df   <*dance> <*life-cycle *marriage> a wedding dance performed by women,
  walking around a large mortar, wielding three long poles as pestles,
  accompanied by men on instruments.
dfx   Twelve to eighteen women walk-dance around the mortar.
  Each woman, when her turn comes, moves in close to
  the mortar and catches a pestle tossed by the preceding woman and plunges
  it into the mortar, then tosses it into the air for the next woman.
```

After she has thrown the third pestle, she steps away and rejoins the circle. Traditional accompaniment is played on a box zither and a raft rattle. More modern accompaniment consists of traditional style tunes played on electric guitar, electric bass, synthesizer, and western drum set.

If additional bands are used for extension bands, they can be listed in a SPECS statement EXTBANDS = in the same format as other band specifications given there, e.g.,

```
EXTBANDS  =  'HIST,COL'
```

The entry above would produce these finderlist entries:

```
.kw   dance
ph   a wedding dance performed by women around a large mortar,
  wielding three long poles as pestles, accompanied by music
  from instruments played by men.::  (+)wadaha

.kw   life-cycle
..kw   marriage
ph   a wedding dance performed by women around a large mortar,
  wielding three long poles as pestles, accompanied by music
  from instruments played by men.::  (+)wadaha

.kw   marriage
xr    life-cycle
```

The contents of both bands, in the above case df and dfx, will appear in the eventually published version, so care should be taken in their wording so as not to be unnaturally redundant.

10. PROPER RECOGNITION OF BRACKETS AND PARENTHESES

(This section has not been written.)

The capability of designating example bands and their associated translation bands from which finderlist keywords may also be extracted, as provided in the previous version, INVERT2, is still available. Everything said about definition bands in this document also applies to translation bands.

other extensions and minor alterations of mark-up symbols

Delimiter changes:

no longer needed for capitalization.

Use instead for font shift back to plain. % and & shifts unchanged

| no longer needed for font shift, now reserved exclusively for keyword-truncation in definition bands

"hook" use _ only, for keyword extension. ~ was legacy of EBCDIC hook.
For a time both ~ and _ were allowed for keyword extension.
Now use only _ . ~ can be used for other purposes, such as "varies with".
Only one character allowed to mean keyword extension.

All these can be reset.

11. MORE WORK FOR THE LEXICOGRAPHER

12. OTHER VOCABULARY DOMAINS

BIBLIOGRAPHY

not yet entered