ENTREZ DIRECT - QUICK TOUR

Entrez Direct (EDirect) is an advanced method for accessing the NCBI's set of interconnected data domains (publication, nucleotide, protein, structure, variation, expression, etc.) from a terminal window. Multi-step queries can be built incrementally, and functions take search terms from command-line arguments. Record retrieval and formatting are normally the final steps in the query.

ENTREZ DIRECT FUNCTION SUMMARY

EDirect consists of a set of Perl scripts that are downloaded to the user's computer. Operations can be grouped into several categories.

Navigation functions support exploration within the Entrez databases:

esearch	Performs a new Entrez search using terms in indexed fields.
elink	Looks up neighbors (within a database) or links (between databases).
efilter	Filters or restricts the results of a previous query.

Retrieval functions return document summaries or downloaded records:

esummary	Obtains document summaries for the query results in XML format.
efetch	Retrieves records or generates reports in a designated format.

Desired fields from XML results can be extracted without writing a program:

xtract	Converts XML into a table of data values.

Several additional functions are also provided:

einfo	Obtains information on indexed fields in an Entrez database.
epost	Uploads lists of unique identifiers or sequence accession numbers.
nquire	Sends a URL request to a web page or CGI service.

ENTERING QUERY COMMANDS

UNIX programs are run by typing the name of the program and then supplying any required or optional arguments on the command line. Argument names are letters or words that start with a dash ("-") character.

In order to begin an Entrez search, the user types "esearch" and then enters the required "-db" and "-query" arguments. For example:

```
  esearch -db pubmed -query "capsaicin cancer pain management"
```

constructs the appropriate Entrez Utilities (EUtils) URL from the query terms and executes the search. EDirect handles many technical details behind the scenes (avoiding the learning curve normally required for EUtils programming), and saves the results on the Entrez history server.

CONSTRUCTING MULTI-STEP QUERIES

The vertical bar ("|") UNIX pipe character connects two programs, sending the output of one to the input of the other. For EDirect functions it communicates the database and history key. Piping the esearch output to an elink command:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | elink -related
```

will look up related articles (precomputed PubMed neighbors) of the initial results. (Using elink -target retrieves associated records between databases.)

PLACING STEPS ON SEPARATE LINES

The same set of commands can be written on multiple lines with the backslash ("\") UNIX line continuation character, for improved readability and easier editing:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | \
  elink -related
```

The list of neighbors can be refined by further term searching in Entrez:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | \
  elink -related | \
  efilter -query "conotoxin NOT morphine"
```

QUALIFYING QUERIES BY INDEXED FIELD

Search terms can be qualified by entering an indexed field abbreviation in brackets. Boolean operators and parentheses can also be used in the query expression for more complex searches.

Some indexed fields from pubmed are:

[ACCN]	Accession	[MAJR]	MeSH Major Topic
[AFFL]	Affiliation	[SUBH]	MeSH Subheading
[ALL]	All Fields	[MESH]	MeSH Terms
[AUTH]	Author	[WORD]	Text Word
[PDAT]	Date - Publication	[TITL]	Title
[FILT]	Filter	[TIAB]	Title/Abstract
[JOUR]	Journal	[UID]	UID

For example:

```
  "Tager H [AUTH] AND glucagon [TIAB]"
```

Indexed fields in the sequence databases include:

[ACCN]	Accession	[ORGN]	Organism
[ALL]	All Fields	[PROP]	Properties
[AUTH]	Author	[PROT]	Protein Name
[FKEY]	Feature Key	[SLEN]	Sequence Length
[FILT]	Filter	[WORD]	Text Word
[GENE]	Gene Name	[TITL]	Title
[KYWD]	Keyword	[UID]	UID

and a sample query in the protein database is:

```
  "alcohol dehydrogenase [PROT] NOT (bacteria [ORGN] OR fungi [ORGN])"
```

The scripting example at the end of this document will produce a report giving the entire set of indexed fields for every Entrez database.

EXAMINING INTERMEDIATE RESULTS

EDirect navigation functions produce a small structured XML object that is passed to the next command. Retrieval functions read the XML values and return the designated report from the history server.

The Count field in the XML intermediate gives the number of records returned by the previous step. A good measure of success is a reasonable (non-zero) value:

```
  <ENTREZ_DIRECT>
    <Db>pubmed</Db>
    <WebEnv>NCID_1_4586177_172.16.22.25_5555_1366675055_388385336</WebEnv>
    <QueryKey>4</QueryKey>
    <Count>10</Count>
    <Step>3</Step>
  </ENTREZ_DIRECT>
```

Checking the result count after each step can help avoid unsuccessful queries.

READING PUBMED REPORTS

Piping the query results to efetch and specifying the "abstract" format:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | \
  elink -related | \
  efilter -query "conotoxin NOT morphine" | \
  efetch -format abstract
```

returns a set of reports that can be read by a person:

```
  1. PLoS One. 2013;8(3):e59293. doi: 10.1371/journal.pone.0059293. Epub ...

  Expression and Pharmacology of Endogenous Cav Channels in SH-SY5Y Human
  Neuroblastoma Cells.

  Sousa SR, Vetter I, Ragnarsson L, Lewis RJ.

  Institute for Molecular Bioscience, The University of Queensland, St. Lucia,
  Australia.

  SH-SY5Y human neuroblastoma cells provide a useful in vitro model to study the
  mechanisms underlying neurotransmission and nociception. These cells are derived
  from human sympathetic neuronal tissue and thus, express a number of the Cav
  channel subtypes essential for regulation of important physiological functions,
  ...
```

Using "efetch -format medline" produces a report suitable for use by common bibliographic management software packages:

```
  PMID- 23536870
  OWN - NLM
  STAT- In-Data-Review
  DA  - 20130328
  IS  - 1932-6203 (Electronic)
  IS  - 1932-6203 (Linking)
  VI  - 8
  IP  - 3
  DP  - 2013
  TI  - Expression and Pharmacology of Endogenous Cav Channels in SH-SY5Y Human
        Neuroblastoma Cells.
  PG  - e59293
  LID - 10.1371/journal.pone.0059293 [doi]
  AB  - SH-SY5Y human neuroblastoma cells provide a useful in vitro model to ...
```

XML DOCUMENT SUMMARIES

EDirect also provides document summaries and other result types that are returned in structured XML format. For example:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | \
  elink -related | \
  efilter -query "conotoxin NOT morphine" | \
  esummary
```

will generate an XML document summary set:

```
  <eSummaryResult>
    <DocumentSummarySet status="OK">
      <DocumentSummary uid="23536870">
        <Id>23536870</Id>
        <PubDate>2013</PubDate>
        <EPubDate>2013 Mar 25</EPubDate>
        <Source>PLoS One</Source>
        <Authors>
          <Author>
            <Name>Sousa SR</Name>
            <AuthType>Author</AuthType>
            <ClusterID>0</ClusterID>
          </Author>
          <Author>
            <Name>Vetter I</Name>
          ...
```

The advantage of XML is that many pieces of information are in specific locations in a well-defined data hierarchy. Assembling individual units of data that are fielded by name, such as:

```
  <PubDate>2013</PubDate>
  <Source>PLoS One</Source>
  <Volume>8</Volume>
  <Issue>3</Issue>
  <Pages>e59293</Pages>
```

requires matching the same general pattern, differing only by the element name. This is much easier than parsing the units from a long, complex string:

```
  1. PLoS One. 2013;8(3):e59293 ...
```

The disadvantage of XML is that data extraction usually requires programming. But EDirect relies on the common pattern of XML value representation to provide a simplified approach to interpreting XML data, as discussed below.

CONVERSION OF XML DATA INTO TABULAR FORM

The xtract function uses command-line arguments to direct the selective conversion of XML data into a tab-delimited table. The -pattern argument divides the results into rows, while placement of data into columns is controlled by -element.

Additional arguments can limit data extraction to specified regions of the XML, filter by data content, and customize the table presentation. These will be introduced in the examples that follow.

Piping the esummary output to:

```
  xtract -outline
```

will give an overview of the XML structure hierarchy:

```
  DocumentSummarySet
    DocumentSummary
      Id
      PubDate
      EPubDate
      Source
      Authors
        Author
          Name
          AuthType
          ClusterID
          ...
```

The outline can help in deciding what arguments to send to xtract. Thus:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | \
  elink -related | \
  efilter -query "conotoxin NOT morphine" | \
  esummary | \
  xtract -pattern DocumentSummary -element Id SortFirstAuthor Title
```

returns the PubMed identifier, first author name, and article title:

```
  23536870  Sousa SR  Expression and Pharmacology of Endogenous Cav Channels ...
  22410003  Vetter I  Characterisation of Na(v) types endogenously expressed ...
  18956616  Fürst Z   [Central and peripheral mechanisms in antinociception: ...
  12566085  Lo YK     Effect of arvanil (N-arachidonoyl-vanillyl-amine), a n ...
  ...
```

INTERACTION WITH UNIX UTILITIES

A tab-delimited table can be processed by many UNIX utiltiies. For example:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | \
  elink -related | \
  efilter -query "conotoxin NOT morphine" | \
  esummary | \
  xtract -pattern DocumentSummary -element Id SortFirstAuthor Title | \
  sort -t $'\t' -k 2,2f -k 3,3f
```

takes the table generated by xtract and sorts first by author name and then (for multiple publications by the same author) alphabetically by title:

```
  11000661  Chiang JS  New developments in cancer pain therapy.
  18956616  Fürst Z    [Central and peripheral mechanisms in antinociception ...
  10864900  Jerman JC  Characterization using FLIPR of rat vanilloid recepto ...
  12566085  Lo YK      Effect of arvanil (N-arachidonoyl-vanillyl-amine), a  ...
  ...
```

Alternatively, a frequency table of title words can easily be generated by passing the titles to a different set of UNIX commands:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | \
  elink -related | \
  efilter -query "conotoxin NOT morphine" | \
  esummary | \
  xtract -pattern DocumentSummary -element Title | \
  sed 's/[^a-zA-Z0-9]/ /g' | tr 'A-Z' 'a-z' | xargs -n 1 | \
  sort | uniq -c | sort -k 1,1nr -k 2,2f
```

This takes the article titles, removes punctuation and accented characters, converts capital letters to lower case, and places each word on a separate line. The words are then sorted alphabetically, occurrence counts for each unique word are calculated, and the results are sorted by frequency:

```
  8 of
  7 in
  4 cells
  4 human
  4 neuroblastoma
  4 sh
  4 sy5y
  3 and
  3 receptor
  2 a
  ...
```

PUBMED ARTICLE XML RECORDS

The PubmedArticle object has a more detailed structure than the DocumentSummary:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | \
  elink -related | \
  efilter -query "conotoxin NOT morphine" | \
  efetch -format xml | \
  xtract -outline
```

and more information is fielded, including author names:

```
  PubmedArticleSet
    PubmedArticle
      MedlineCitation
        PMID
        DateCreated
          Year
          Month
          Day
        Article
          Journal
            ISSN
            JournalIssue
              Volume
              Issue
              PubDate
                Year
            Title
            ISOAbbreviation
          ArticleTitle
          Pagination
            MedlinePgn
          ELocationID
          Abstract
            AbstractText
          Affiliation
          AuthorList
            Author
              LastName
              ForeName
              Initials
            ...
```

Using this information to craft a new xtract statement:

```
  esearch -db pubmed -query "capsaicin cancer pain management" | \
  elink -related | \
  efilter -query "conotoxin NOT morphine" | \
  efetch -format xml | \
  xtract -pattern PubmedArticle -element "MedlineCitation/PMID" LastName
```

results in a table of all authors for each record:

```
  23536870  Sousa   Vetter  Ragnarsson  Lewis
  22410003  Vetter  Mozar   Durek       Wingerd  Alewood  Christie  Lewis
  18956616  Fürst
  12566085  Lo      Chiang  Wu
  ...
```

(Note that -element "MedlineCitation/PMID" uses the parent/child construct to prevent picking up PMIDs that occur in the CommentsCorrections objects. The -first or -last arguments can be used instead of -element, if appropriate.)

EXPLORATION OF SETS WITHIN XML

Individual PubmedArticles can be retrieved directly by efetch:

```
  efetch -db pubmed -id 1937004 -format xml
```

The resulting XML may contain a list of Medical Subject Headings (MeSH Terms):

```
  ...
  <MeshHeadingList>
    <MeshHeading>
      <DescriptorName>Adenosine Triphosphatases</DescriptorName>
    </MeshHeading>
    <MeshHeading>
      <DescriptorName>Amino Acid Sequence</DescriptorName>
    </MeshHeading>
    <MeshHeading>
      <DescriptorName>Base Sequence</DescriptorName>
    </MeshHeading>
    ...
```

Visiting each MeSH term with a -block statement, and customizing the output format (changing the tab between fields to a newline character):

```
  efetch -db pubmed -id 1937004 -format xml | \
  xtract -pattern PubmedArticle -tab "\n" -element "MedlineCitation/PMID" \
    -block MeshHeading -tab "\n" -element DescriptorName
```

produces a list of MeSH terms, one per line, following the PubMed ID:

```
  1937004
  Adenosine Triphosphatases
  Amino Acid Sequence
  Base Sequence
  ...
  Recombination, Genetic
  Saccharomyces cerevisiae
  Saccharomyces cerevisiae Proteins
```

NESTED EXPLORATION OF XML SUBREGIONS

MeSH terms can have one or more subheadings:

```
    ...
    <MeshHeading>
      <DescriptorName>Recombination, Genetic</DescriptorName>
      <QualifierName>genetics</QualifierName>
    </MeshHeading>
    <MeshHeading>
      <DescriptorName>Saccharomyces cerevisiae</DescriptorName>
      <QualifierName>genetics</QualifierName>
      <QualifierName>radiation effects</QualifierName>
    </MeshHeading>
    <MeshHeading>
      <DescriptorName>Saccharomyces cerevisiae Proteins</DescriptorName>
    </MeshHeading>
  </MeshHeadingList>
  ...
```

Adding a -subset statement within the -block allows nested exploration of the subheadings for each MeSH term:

```
  efetch -db pubmed -id 1937004 -format xml | \
  xtract -pattern PubmedArticle -tab "" -element "MedlineCitation/PMID" \
    -block MeshHeading -pfx "\n" -tab "" -element DescriptorName \
      -subset QualifierName -pfx "/" -tab "" -element QualifierName
```

and results in a list of MeSH terms and associated subheadings:

```
  1937004
  Adenosine Triphosphatases
  Amino Acid Sequence
  Base Sequence
  ...
  Recombination, Genetic/genetics
  Saccharomyces cerevisiae/genetics/radiation effects
  Saccharomyces cerevisiae Proteins
```

RECORDING VALUES IN VARIABLES

A value can be recorded in a variable and then displayed multiple times as needed. Variables are indicated by a hyphen followed by a string of capital letters or digits. The variable "-PMID" is referred to as "&PMID" in an -element argument.

For example:

```
  efetch -db pubmed -id 1937004 -format xml | \
  xtract -pattern PubmedArticle -PMID "MedlineCitation/PMID" \
    -block MeshHeading -tab "\n" -element "&PMID",DescriptorName
```

produces a list of MeSH terms, with the PMID in the first column of each row:

```
  1937004  Adenosine Triphosphatases
  1937004  Amino Acid Sequence
  1937004  Base Sequence
  ...
  1937004  Recombination, Genetic
  1937004  Saccharomyces cerevisiae
  1937004  Saccharomyces cerevisiae Proteins
```

CONDITIONAL PROCESSING

Xtract provides arguments that filter by data content (-match and -avoid) or elements (-present and -absent). Parallel statements are used to handle alternative conditions. For example:

```
  efetch -db pubmed -id 1937004 -format xml | \
  xtract -pattern PubmedArticle -PMID "MedlineCitation/PMID" \
    -group MeshHeading \
      -block MeshHeading -present QualifierName \
        -subset DescriptorName -TERM "DescriptorName" \
        -subset QualifierName -tab "\n" -element "&PMID","&TERM",QualifierName \
      -block MeshHeading -absent QualifierName \
        -subset DescriptorName -tab "\n" -element "&PMID",DescriptorName
```

has separate sections for MeSH terms with and without subheadings, and produces:

```
  1937004  Adenosine Triphosphatases
  1937004  Amino Acid Sequence
  1937004  Base Sequence
  ...
  1937004  Recombination, Genetic             genetics
  1937004  Saccharomyces cerevisiae           genetics
  1937004  Saccharomyces cerevisiae           radiation effects
  1937004  Saccharomyces cerevisiae Proteins
```

XML ATTRIBUTES

The MeSH term and subheading fields actually have major topic attributes:

```
  <MeshHeading>
    <DescriptorName MajorTopicYN="N">Saccharomyces cerevisiae</DescriptorName>
    <QualifierName MajorTopicYN="Y">genetics</QualifierName>
    <QualifierName MajorTopicYN="N">radiation effects</QualifierName>
  </MeshHeading>
```

that can be selected as "DescriptorName@MajorTopicYN" or simply "@MajorTopicYN":

```
  efetch -db pubmed -id 1937004 -format xml | \
  xtract -pattern PubmedArticle -PMID "MedlineCitation/PMID" \
    -group MeshHeading \
      -block MeshHeading -present QualifierName \
        -subset DescriptorName -TERM "DescriptorName" -MAJR "@MajorTopicYN" \
        -subset QualifierName -tab "\n" \
          -element "&PMID","&TERM","&MAJR",QualifierName,"@MajorTopicYN" \
      -block MeshHeading -absent QualifierName \
        -subset DescriptorName -tab "\n" \
          -element "&PMID",DescriptorName,"@MajorTopicYN"
```

The results are suitable for importing into a database or spreadsheet program:

```
  1937004  Adenosine Triphosphatases          N
  1937004  Amino Acid Sequence                N
  1937004  Base Sequence                      N
  ...
  1937004  Recombination, Genetic             N  genetics           N
  1937004  Saccharomyces cerevisiae           N  genetics           Y
  1937004  Saccharomyces cerevisiae           N  radiation effects  N
  1937004  Saccharomyces cerevisiae Proteins  Y
```

EXPLORING SEPARATE XML REGIONS

Multiple -block statements can be used in a single xtract to explore different areas of the XML. Combining fields with commas allows them to be treated as sets, and the tab that normally separates these can be replaced with a -sep argument:

```
  efetch -db pubmed -id 781293,2678811,6301692,8332518 -format xml | \
  xtract -pattern PubmedArticle -element "MedlineCitation/PMID" \
    -block AuthorList -sep "/" -element LastName "#Author" \
    -block PubDate -sep " " -element Year,Month MedlineDate \
    -block DateCreated -sep "-" -element Year,Month,Day | \
  sort -t $'\t' -k 3,3n | column -s $'\t' -t
```

produces a table that allows easy parsing of author names, counts the number of authors present, and prints the date each record was published and the date it was entered into PubMed, sorting the results by the computed author count:

```
  781293   Casadaban                         1  1976 Jul      1976-10-02
  6301692  Krasnow/Cozzarelli                2  1983 Apr      1983-06-17
  8332518  Benson/Lipman/Ostell              3  1993 Jul      1993-08-17
  2678811  Mortimer/Schild/Contopoulou/Kans  4  1989 Sep-Oct  1989-11-22
```

(The PubDate object can exist either in structured or string form, but would not contain a mixture of both types, so "-element Year,Month MedlineDate" will only contribute a single column to the output.)

STORING COMMON PHRASES IN ALIAS FILES

Frequently used, long, or complicated search phrases can be saved in a file to avoid having to retype (or copy and paste) the full text for each query. Each line of the file has a shortcut keyword, a tab character, and the expanded search term. Shortcuts are referenced by placing them in parentheses and prefixing with a pound ("#") sign. For example, given a file named "query_aliases" containing:

```
  jour_filt    [MULT] AND ncbijournals [FILT]
  trans_imm    (transposition OR target) immunity
```

the esearch line in:

```
  esearch -alias query_aliases -db nlmcatalog -query "Science (#jour_filt)" | \
  esummary | \
  xtract -pattern DocumentSummary -element ISOAbbreviation \
    -subset ISSNInfo -sep "|" -element issn,issntype | \
  column -s $'\t' -t
```

will be expanded to:

```
  esearch -db nlmcatalog -query "Science [MULT] AND ncbijournals [FILT]"
```

with the query producing:

```
  J. Zhejiang Univ. Sci.  1009-3095|Print  1009-3095|Linking
  Science (80- )          0193-4511|Print  0193-4511|Linking
  Science                 0036-8075|Print  1095-9203|Electronic  ...
```

SEQUENCE RECORDS IN INSDSEQ XML

Sequences shown in GenBank or GenPept flatfile format have features that describe our current understanding of the biology encoded in that region.

Sequence records can be retrieved in an XML version of the GenBank or GenPept flatfile. Feature and qualifier names are data values, not XML tags, and require -match to select the desired object. A query for snail venom mature peptides:

```
  esearch -db protein -query "conotoxin AND mat_peptide [FKEY]" | \
  efetch -format gpc -mode xml | \
  xtract -pattern INSDSeq -ACCN INSDSeq_accession-version \
    -group INSDFeature -match ">mat_peptide<" \
        -avoid "<INSDFeature_partial" -pfx "\n" -element "&ACCN" \
      -block INSDQualifier -match ">peptide<" -element "%INSDQualifier_value" \
      -block INSDQualifier -match ">product<" -element INSDQualifier_value \
      -block INSDQualifier -match ">peptide<" -element INSDQualifier_value | \
  grep conotoxin | sort -t $'\t' -u -k 3,4 | \
  sort -t $'\t' -k 2,2n | column -s $'\t' -t
```

calculates the length of each mature peptide, and prints the product name and peptide sequence, removing redundant entries and sorting by peptide length:

```
  ADB43130.1  15  conotoxin Cal 1a       KCCKRHHGCHPCGRK
  ADB43131.1  15  conotoxin Cal 1b       LCCKRHHGCHPCGRT
  AAO33169.1  16  alpha-conotoxin GIC    GCCSHPACAGNNQHIC
  ADB43127.1  16  conotoxin Cal 5.1      DPAPCCQHPIETCCRR
  AAD31913.1  18  alpha A conotoxin Tx2  PECCSHPACNVDHPEICR
  ...
```

(The sequence accession is captured in a variable for use with each mat_peptide in the record. Prefix substitution ensures that every peptide is shown on a separate line. This also works for multi-product precursor proteins such as proinsulin.)

(Incorporating fragments of the surrounding XML tags in -match arguments prevents coincidental matches to free text in note qualifiers.)

Multiple -avoid or -match conditions are specified with -and and -or commands:

```
  -group INSDFeature -avoid ">proprotein<" -and ">sig_peptide<" \
    -block INSDQualifier -match ">calculated_mol_wt<" -or ">peptide<" \
```

AUTOMATION WITH SCRIPTS

Taking an adventurous plunge into the world of programming, a shell script can be written when each output line of one step needs to be processed independently, instead of output being piped in its entirety to the next command. (The simplest shell script can be merely a copy of the set of commands that are typed into the terminal for execution.)

In shell scripts, variables can be set to the results of a command by enclosing the statements in backtick ("`") characters. The variable name is prefixed by a dollar sign ("$") to use its value as an argument in another command. Comments start with a pound sign ("#") and are ignored. Quotation marks within quoted strings are "escaped" with a backslash ("\"). Subroutines can be used to collect common code or simplify the organization of the script.

For example:

```
  #!/bin/bash -norc

  parse_fields() {
    echo "$1" | \
    xtract -pattern Field -element Name FullName Description | \
    sort -t $'\t' -k 2,2f | \
    perl -pe 's/(.+?)\t(.+?)\t(.+)/\[$1\]\t$2\t$3/' | \
    column -s $'\t' -t
  }

  dbs=`einfo -dbs | xtract -pattern DbName -element DbName | sort`

  for db in $dbs; do
    eix=`einfo -db $db`

    echo "$db"
    echo ""
    flds=`parse_fields "$eix"`
    echo "$flds"
    echo ""

    sleep 1
  done
```

will obtain the list of Entrez databases, and then return the abbreviations, names, and descriptions of indexed search fields, for each individual database:

```
  ...
  epigenomics

  [ACCN]  Accession      Accession number of sequence
  [ALL]   All Fields     All terms from all searchable fields
  [AUTH]  Author         Author
  [CDAT]  Create Date    CreateDate
  [DOCT]  Document Type  DocType
  [FILT]  Filter         Limits the records
  [KYWD]  Keyword        Keyword
  [ORGN]  Organism       scientific and common names of organism
  [PRID]  Project ID     ProjectId
  [TXID]  Taxonomy ID    TaxId
  [WORD]  Text Word      Text
  [TITL]  Title          Title
  [UID]   UID            Unique number assigned to publication
  [MDAT]  Update Date    UpdateDate
  ...
```

Creative use of the "sh" and "xargs" commands can obtain the same behavior from the command line, without the need to write separate script files. Within the "sh" command string, the database argument passed by xargs is substituted at the "$0" variable. All of the sh commands are run separately on each database.

Because the sh instructions are enclosed between single quote ("'") characters (apostrophes), in-line perl commands cannot be used, nor can sort be told to use tabs (instead of tabs or spaces) to delimit columns. But a combination of xtract customization and the trick of temporarily replacing spaces with asterisks:

```
  einfo -dbs | xtract -pattern DbName -element DbName | sort | \
  xargs -n 1 sh -c 'einfo -db "$0" | \
    xtract -pattern DbInfo -tab "\n\n" -element DbName \
      -block Field -pfx "[" -sep "]\t" -tab "\n" -element Name,FullName | \
    sed "s/ /*/g" | sort -k 2,2f | sed "s/*/ /g" | expand'
```

allows the desired sorting by full field name and produces results that are alphabetized for each database:

```
...
pubmed
[AFFL]  Affiliation
[ALL]   All Fields
[AUTH]  Author
[COLN]  Author - Corporate
[FAUT]  Author - First
[FULL]  Author - Full
[LAUT]  Author - Last
[AUCL]  Author Cluster ID
[BOOK]  Book
[CDAT]  Date - Completion
[CRDT]  Date - Create
[EDAT]  Date - Entrez
[MHDA]  Date - MeSH
[MDAT]  Date - Modification
[PDAT]  Date - Publication
[ECNO]  EC/RN Number
[ED]    Editor
[EPDT]  Electronic Publication Date
[EID]   Extended PMID
[FILT]  Filter
[GRNT]  Grant Number
[INVR]  Investigator
[FINV]  Investigator - Full
[ISBN]  ISBN
[ISS]   Issue
[JOUR]  Journal
[LANG]  Language
[LID]   Location ID
[MAJR]  MeSH Major Topic
[SUBH]  MeSH Subheading
[MESH]  MeSH Terms
[OTRM]  Other Term
[PAGE]  Pagination
[PAPX]  Pharmacological Action
[CNTY]  Place of Publication
[PPDT]  Print Publication Date
[PTYP]  Publication Type
[PUBN]  Publisher
[PID]   Publisher ID
[SI]    Secondary Source ID
[SUBS]  Supplementary Concept
[WORD]  Text Word
[TITL]  Title
[TIAB]  Title/Abstract
[TT]    Transliterated Title
[UID]   UID
[VOL]   Volume
...
```