The Unix POP-2 System

William Clocksin
Department of Artificial Intelligence
University of Edinburgh
Edinburgh, Scotland
June, 1979

Software Report 4

A POP-2 system is available for use with the Unix operating system. The system is intended to provide the standard words of Reference Manual POP-2, and many of the enhancements implemented in Revised POP-2 ('WonderPOP') are available. The ASCII character set is used, so the format of text items has been brought up to date. The "popmess" facilities have been reduced because it is possible to call the Unix Shell from within POP-2. A selection of libraries is available. The system occupies 8K words of pure (shared) code and 2K words of per-process data, leaving 22K words for user workspace.

If you produce a thesis, article, or report as a result of work that involved this system in any way, a reference to this documentation must appear.

Format of Text Items

Characters

Characters are represented internally in the standard 7-bit ASCII code. This allows upper and lower case letters to be distinguished, and more characters are available for use as signs. Characters are always stored in an 8-bit format. All eight bits are available to the user, providing the potential for a 256 character set.

Each character in the ASCII code belongs to one and only one 'character class'. The user is permitted to change the class to which a character belongs. The classes and their standard members are:

Letters: Upper and lower case letters are distinguished
Digits: 0 1 2 3 4 5 6 7 8 9
Signs: $ * : = - \ @ ^ ~ + < > ? / & |
Bridges: (underline) (backspace)
Separator: " , ; .
Comment Bracket: !
Opening Bracket: ([{
Closing Bracket:)] }
Decorator: %
String Quote: '
Alphabeticiser: `
Code Quote: #
Blank Space: (tab) (linefeed) (return) (formfeed) (space)

The remaining non-printing control characters are treated as

separators. Separators do not bind with other characters, so they are read by item repeaters as separate items.

The decorator is treated as a sign that binds with brackets to form decorated brackets. In any other context however, it is treated as a sign.

The alphabeticiser indicates that the character following it is treated as a letter, regardless of what the character's class is. The code quote, also followed by a character, reads as the number which is the ASCII code for the character. For example, #a reads in as octal 141 but is clearer in its intent.

Typing certain control characters may cause action to be taken in accord with Unix conventions. These are ^q, ^w, ^r, ^t, ^o, ^s, ^z, ^\, and ESC. Control-c is detected by POP-2, so Unix interrupt is disabled. Unix 'quit' may be used as Unix intends.

Numbers

Integers are in the range -32768 to +32767. Reals are in the range 1.0e-34 to 1.0e34 roughly. Any number may be prefixed by a digit and a colon, meaning the number is expressed in the radix indicated by the digit. The digit cannot be 0. Examples: (8:176 5:2.3). Reals may be expressed in exponential notation. The exponent is introduced with the character 'e'. Examples: (12.4e5 6.02e-23 0.2e+17). At least one digit must appear on both sides of the decimal point.

Words

Words read by an item repeater may consist solely of (a) letters, digits, and bridges; or (b) signs and bridges. In (a), the first character must be a letter. In (a) and (b), bridges cannot be the first character of a word. Bridges can enable more readable identifiers to be used, such as end_of_line or lower_bound. The backspace can be used (with caution!) to make composite overstruck words. Such words are not effective when printed on most display screens.

Strings

Strings are typed as characters between string quotes. There is only one string quote, which both opens and closes the string. A string quote character is incorporated within a string by preceding it with another string quote.

Comments

Comments are typed as characters between comment quotes. There is only one comment quote, which both opens and closes the comment. Comments are ignored by item repeaters, but may be read by character repeaters.

File Specifications and Libraries

Some arguments to POP-2 functions are used to specify file names.

The Unix conventions for path names are observed. A file specification for POP-2 is a list containing the path name. Examples: ([alpha] [/bin/foo] [/usr/lib/prog.c]). Library programs are specified by using a file specification with the word "lib" followed by a library name. For example, the array processing functions are held in the library named "arrays". To make these functions available, compile([lib arrays]). A library specification can be used wherever a file specification can. The list of what library programs are available is included in this document.

Using POP-2

To use POP-2, type the Unix command 'pop2'. Arguments are permitted on the command line in the usual way. If a file named '.pop' appears on the current directory, it is compiled first automatically when the 'pop2' command is invoked. If arguments were given on the command line, the designated files are then compiled in the order specified. If an argument denotes a file that does not exist, a warning message is printed and the subsequent arguments are ignored. Finally, the system calls "setpop" to begin the interactive session.

POP-2 may be used as a Unix filter. Therefore, typing a 'control-z' character will cause end of file on the user's terminal, and will cause an exit from POP-2.

If at any time the interrupt character (^c or "break") is typed, the function assigned to "popbreak" will be executed. When the system is entered, popbreak is initialised to setpop. However, any function can be assigned to popbreak to produce any desired effect when ^c is typed. For example, the library [lib breaks] uses popbreak to provide facilities for interrupting and resuming levels of interaction. To exit from POP-2 under control of a program, use popmess([exit]).

When typing, a 'control-g' followed by 'return' will enter setpop. Setpop unwinds the activation record stack so that local variables are unbound from each function activation occurring previous to the setpop call.

There are some files of interest in the directory named '/pop'. The file /pop/mods is a file of all the changes that have been made to the system at various dates. The files matching /pop/doc/doc* are roff files that produce this documentation. The /pop/lib directory contains the source code and documentation for the library packages that are accessed by [lib] specifications.

Standard Words Available in Unix POP-2

All of the predefined standard words, denoting functions, operations, macros, and variables, are listed below together with a brief description. All words are (I hope) implemented as described in the Grey Book (Programming in POP-2; Burstall, Collins, and Popplestone; Revised edition, 1977), except where legend characters appear before a word. The legends are:

N - New feature, not appearing in the Grey Book.
R - Revised to a better definition.
X - Not implemented. To be implemented in due course.
O - Obsolete feature, which has been overtaken by events.

Where possible, the arguments of functions are denoted by lowercase letters. Optional arguments are enclosed in {} brackets.

N abs(n)
Returns the absolute value of n.

R and
As an improvement over the Grey Book definition, "and" and "or" may appear in any expression. They evaluate only enough of their arguments to determine the result. The left-hand argument is evaluated and the result put on the stack. If the result is non-zero, the result is erased and the result of evaluating the right-hand argument is put on the stack. Otherwise, the right-hand argument is not evaluated.

X appdata(x,f)
Applies function f to successive components of the data structure x.

applist(l,f)
Applies function f to successive components of the list l (see appdata, maplist).

apply(f)
Applies function f.

X appstate(f)
Creates a saved state for the currently active barrier and applies f to it. User assignable.

arctan(n)
Returns the inverse tangent of n radians.

atom(x)
Returns true if item x is not a pair, otherwise false (see back, conspair, destpair, front, ispair).

back(p)
A doublet to select and update the back of the pair p (see atom, back, conspair, destpair, ispair).

X barrierapply(a1,a2,...,ai,f,i)
Creates a barrier and applies function f to the i specified arguments a1,a2,...ai. (see appstate, issavedstate,reinstate).

O booland(x,y)
This function is superseded by "and", which can now be used in expressions.

O boolor(x,y)
Superseded by "or", which can now be used in any expression.

boundslist(a)
Returns a list of low and high bounds for the array a (see newanyarray, newarray). To use, compile [lib arrays].

N break
Syntax word -- causes an exit from the surrounding "while" or "until" loop (see "until", "while", "continue").

cancel x y ... z;
Syntax word - breaks the association between the identifiers x, y, ...,z and their current variables and syntactic properties. A cancellation has effect only within the section it is used.

charin()
The character repeater for the standard input device (user's terminal). The result is either a 7-bit integer or termin. integer or termin.

charout(c)
The character consumer for the standard output device (user's terminal). The argument c is either a 7-bit integer or termin.

close
Syntax word: ends a "forall", "if", "unless", "while", or "until".

O comment ... ;
Syntax word - reads and ignores text items from proglist until ";" has been read. Superseded by the comment quote character.

compile(x)
Compiles the text read from the file specification x. (see cucharin, popval).

cons(x,l)
Constructs a list with head x and tail l (see dest, hd, islist, nil, null, tl).

conspair(x,y)
Constructs a pair with front x and back y (see atom, back, destpair, front, ispair).

consref(x)
Constructs a reference with contents x (see cont, destref, isref).

consword(c1,c2,...,ci,i)
Constructs a word having the i characters c1 through ci (See charword, destword, isword).

cont(r)
A doublet that selects and updates the contents of the reference r (see consref, destref, isref).

N continue
Syntax word -- causes the current iteration of the enclosing "while" or "until" loop to be abandoned, and the next iteration started (see "while", "until", "break").

X copy(x)
Returns a top level copy of the data structure x.

copylist(l)
Returns a top level copy of the list l.

cos(n)
Returns the cosine of n.

cucharin()
Local to compile holding the character repeater. User assignable.

cucharout(x)
The standard character output function for the standard output device. Used by nl, sp, syspr.

datalength(x)
Returns the number of components in data structure x.

X datalist(x)
Returns a list of the components of datastructure x.

dataword(x)
Returns the word associated with the data class of x.

dest(l)
Returns the head and the tail of the list l (see cons, hd, islist, nil, null, tl).

destpair(p)
Returns the front and back of the pair p (see atom, back, conspair, front, ispair).

0 destref(r)
Returns the contents of the reference r (see consref, cont, isref). Use "cont" instead. It does the same thing.

destword(w)
Returns the characters of the word w and the count of the number of characters (see charword, consword, isword).

N edit(l)
Suspends POP-2 and enters the EM editor with the file specification as the argument. When the edit is finished, POP-2 is resumed from the point where "edit" was called.

else
Syntax word - introduces an alternative condition (cl close, elseif, if, unless).

elseif
Syntax word - introduces an alternative condition (see close, else, if, unless).

end
Syntax word - ends the definition of a function body (see function, lambda, macro, operation).

endsection
Syntax word - ends a section and returns to the surrounding environment.

X equal(x,y)
Returns true if the items x and y are identical; otherwise, if x and y belong to the same data class and have the same number of components, and corresponding components of one are equal to those of the other, the result is again true. Otherwise the result is false.

erase()
Removes the top item from the user stack.

errfun(x,y)
Called by the system when an error occurs. User assignable, initialised to syserr. The argument x is the culprit, and y is the error number (see syserr).

exit
Syntax word - causes a jump to the end of the current function and ends the current conditional (see close, else, elseif, if, return, unless).

exp(n)
Returns the exponential function of n.

false
Truthvalue with value 0 (see true).

fnpart(f)
A doublet that selects or updates the function that was partially applied to form the closure f (see frozval, isclosure,partapply, (%%)).

fnprops(f)
A doublet that selects and updates the item associated with the function f (see isfunc, updater). The fnprops of a standard function may not be updated.

fntolist(f)
Returns a dynamic list from the repeater function f (see islink).

forall
A macro providing primitive loop facilities.

front(p)
A doublet that selects or updates the front component of the pair p (see atom, back, conspair, destpair, ispair).

frozval(i,f)
A doublet that selects or updates the ith frozen formal of the closure f (see fnpart, isclosure, partapply, (%%)).

function w {a b ... c} {=> x y ... z}
Syntax word - introduces a named function definition (see end,

lambda, macro, operation).

N gc_gag
A variable that controls the printing of a message at eah call of the garbage collector. Set gc_gag to 1 to inhibit messages.

goon
Syntax word - terminates the evaluation of text by popval when read from a list (see popval).

goto
Syntax word - followed by a label, causes control to pass to the named labelled statement (see return, :). Unfashionable.

hd(l)
A doublet that selects or updates the head of list l (see cons, dest, islist, nil, null, tl).

identfn()
The identity function.

identprops(w)
Returns information about the syntactic properties of the word w when used as an identifier.

R if ... then ... close;
Syntax word - introduces a conditional. Condtionals are permitted at execute level. (See "unless", "while", "until")

incharitem(x)
Returns an item repeater function from the character repeater function.

init(i)
Constructs a standard full item strip of i components (see isstrip, subscr).

initc(i)
Constructs a standard packed-byte strip of size 8 with i components (see iscstrip, subscrc, ').

intof(n)
Returns thr integer part of the argument, which may not be the integer must closely approximating the argument.

O iscompnd(x)
Returns true if the item x is not a simple item, that is if x is not a number, otherwise false. Use not(isnumber(x)).

isfunc(x)
Returns true if x is a function, including arrays and closures, otherwise false.

isinteger(x)
Returns true if x is an integer, otherwise false.

X islink(x)
Returns true if x is a link, otherwise false.

islist(x)
Returns true if x is a list, otherwise false.

isnumber(x)
Returns true if x is a number, real or integer, otherwise false.

O ispair(x)
Returns true if x is a pair, otherwise false. Use not(atom(x)).

isreal(x)
Returns true if x is a real number, otherwise false.

isref(x)
Returns true if x is a reference, otherwise false.

isstrip(x)
Returns true if x is a strip or a string, otherwise false.

isword(x)
returns true if x is a word, otherwise false.

itemread()
Reads the next text item from proglist, that is it will read a word, character strip, or unsigned number (see listread, numberread).

jumpout(f,i)
When called within the function g, produces a function, which, when called anytime before returning from g, obeys f and takes an immediate exit from g with the top i items of the user stack added to the stack as it was on entry to jumpout.

lambda
Syntax word - introduces an unnamed function definition (see end, function, macro, operation).

length(l)
Returns the length of the list l.

listread(l)
Reads the next list constant from proglist (see itemread,numberread, proglist).

log(n)
Returns the logarithm of n

logand(x,y)
Returns the logical and of the representations of x and y. The result is a 16-bit pattern, a signed integer (see lognot, logor, logshift).

lognot(x)
Returns the logical not of the representation of x. The result is

a 16-bit pattern, a signed integer (see logand, logor, logshift).

logor(x,y)
Returns the logical or of the representation of x and y. The result is a 16-bit pattern, a signed integer (see logand, lognot, logshift).

logshift(x,i)
Shifts the bit pattern of x logically the number of places specified by the integer i. The result is a 16-bit pattern, a signed integer. (see logand, lognot, logor).

O loopif
Syntax word - introduces a conditional loop. Superceded by "while" and "until".

macresults(l)
The results of all calls of macresults within a macro are appended, and the text in the resulting list replaces the macro call.

macro
Syntax word - (1) recognised as a declaration type in a vars statement. (2) introduces a function with macro identifier (see function , lambda, operation).

maplist(l,f)
Constructs a list from the results of applying the function f to each element of the list l (see applist). Maplist is not defined as [%applist(l,f)%], so arbitrarily long lists can be constructed.

meaning(x)
Is a permanent property doublet which associates an item with the item x (see appdic, newprop).

newanyarray(l,f,fi,fd)
Returns an array given a list l of low and high bounds, an initialisation function f, and strip initiator function fi and selector/updater doublet fd (see boundslist, newarray). The definitions for newanyarry, newarray, and boundlist are written in POP-2, and are made available by compiling [lib arrays].

newarray(l,f)
Returns an array of full items given a list l of low and high bounds and an initialisation function f (see boundslist, newanyarray). Same as the function returned by partially applying newanyarray to init and subscr. To use, compile [lib arrays].

N newprop()
Returns a property doublet allowing an item to be associated with any other. Example: newprop() -> colour; "red" -> colour("flag"); colour("flag")=> (prints red). The property test is item equality using "=". The "meaning" function was defined by using "newprop".

N newvars_gag
A variable that controls the printing of a message whenever an

undeclared variable is encountered bty the compiler. The variable becomes declared automatically. Set newvars_gag to 1 to inhibit the message, 0 to allow it.

X nextchar(f)
Returns the next character that would be read by the item repeater function f. If f is itemread, proglist is checked and, if suitable, the next character that would be read to solidity proglist is returned.

nil
Is the constant null list (see cons, dest, hd, islist, null, tl).

nl(i)
Outputs a carriage-return followed by i line-feed characters to the current output device (see cucharout, pr, sp, syspr).

nonmac m
Syntax word - causes the following macro identifier to be read, without macro expansion, as an untyped identifier (see nonop).

nonop o
Syntax word - causes the following operation identifier to be read, without the function being obeyed, as an untyped identifier (see nonmac).

not(x)
Returns true if x is false, otherwise false. (see booland, boolor)

null(l)
Returns true if l is the null list, otherwise false if l is a list (see cons, dest, hd, islist, nil, tl).

numberread()
Reads a signed number from proglist (see impread, itemread, listread).

operation i
Syntax word - (1) followed by an integer precedence recognised as a declaration type in a vars statement. syntax word - (2) followed by integer precedence introduces a function with operation identifier (see function, lambda, macro).

R or
As an improvement over the Grey Book, "and" and "or" can be used in any expression. They evaluate only as many arguments as needed to determine the result. The right-hand argument is evaluated and the result is put on the stack. If the result is zero, the result is erased and the result of evaluating the left-hand argument is put on the stack. Otherwise, the left-hand argument is not evaluated.

partapply(f,l)
Returns the closure formed by partially applying f to the items of the list l (see fnpart, frozval, (%%)).

N popbreak()
The function called whenever the interrupt key (^c or "break") is pressed. Initialised to setpop. Not changed at setpop.

N popcaller(i)
Returns the ith nested function calling popcaller, where popcaller(0) is the function within which popcaller is called. If i>0, then popcaller(0) is done. If there is no popcaller, then 0 is returned.

NX popchar(c)
Returns or updates the character class (an integer code) of the character whose ASCII code is c.

N popfile
An identifier whose value is the file specification currently being compiled, or 0 if no file is being compiled (see popline).

N pop_gc()
Explicitly reclaims unused memory space. Unused space is recovered automatically by the system. The only use of pop_gc is in conjunction with popword to determine how many words are used by the current workspace.

NX popline(f)
Returns the number of newline characters that have been returned by the character repeater f since it was constructed. Used by errfun to determine on what line of a file an error occurred.

popmess(l)
Gives a means of controlling files and other non-language facilities specified within the list l.

N poppip(in,out)
The two arguments are a repeater and consumer, in that order. "poppip" repeats and consumes until an end-of-file is reached on the consumer, thus copying from one to another. If a list is given as an argument, "poppip" will assume it is a file specification, and will convert it to the appropriate repeater or consumer.

R popval(l)
Interprets the list l as an imperative sequence and executes it. It is terminated by either the end of the list being reached or the word goon being read (see compile, goon).

N popword()
Returns the number of machine words occupied by the current workspace.

pr(x)
The standard function for printing any item.

proglist
The current list of text items from which popval and itemread take items. User assignable.

prreal(n,i,j)
Outputs n to the current output device as a real number with i places before and j places after the decimal point.

0 prstring(s)
Has been removed from the Revised Standard POP-2. "pr" now prints strings without quotes.

realof(n)
Converts n to a real number and returns the result (see intof).

recordfns(x,l)
Creates a new class of record cells with dataword x and components given by l, the list of field sizes. The results are the constructor and destructor functions and a selector doublet for each component (see stripfns). The "compnd" field size is not used. Use 0 instead.

X reinstate(s)
Returns control to immediatly past the point at which the saved-state s was created by appstate (see appstate, barrierapply).

return
Syntax word - causes an exit from the current function (see "exit").

rev(l)
Returns a list of the elements of the list l in the reverse order.

samedata(x,y)
Returns true if x and y both belong to the same data class, otherwise false.

section s {x y ...z} {=> a b ... c};
Syntax word - allows portions of a program to be written in a manner which protects against identifier clashes with other programs. The scope of a section is terminated by endsection.

setpop()
Causes an exit to be taken from all currently active functions and the user stack to be cleared. The error reporter and current output devices are reset to their standard values. Text is then read and executed from the standard input (see charin, proglist, popval)

sign(n)
Returns -1 if n is negative, 0 if n is zero, and +1 if n is positive.

sin(n)
Returns the sine of n radians.

sp(i)
Outputs i spaces to the current output device (see cucharout, nl, pr).

sqrt(n)
Returns the square root of n.

stacklength()
Returns the number of items currently on the user stack, excluding the result itself.

stripfns(x,y)
Constructs the initiator function and selector doublet for a new strip class with dataword x and component size x. The size may be 0 for full item components, or an integer between 1 and 15 for packed byte components. (see recordfns)

subscr(i,s)
A doublet that selects or updates the ith component of the strip s constructed by init (see init, isstrip).

subscrc(i,s)
A doublet that selects or updates the ith component of the character strip s constructed by initc (see init, iscstrip).

syserr(x,y)
Calls the standard error reporter for error y with culprit x. If the error is during execution, the name of the executing function and the calling sequence are printed. If the error is during compilation, the function name, section name, file name, and file line number are printed if appropriate.

termin
The value of the terminator for a file.

tl(l)
A doublet that selects or updates the tail of list l.

then
Syntax word - terminates a conditional test introduced by if, elseif, while, unless, or until.

true
Truthvalue with value 1 (see false).

undef
A system identifier with value "undef". The value of any newly declared variable is undef.

N unless ... then ... close;
Syntax word - introduces a conditional satisfied when false. unless ... then ... is equivalent to if not(...)then ... Condtionals are permitted at execute level. (See "if", "while", "until").

N until ... then ... close;
Syntax word - introduces a conditional loop. The following expression sequence is evaluated. If it is false, the imperitive sequence between "then" and "close" is evaluated and the condtion is tested again (see "while"). Condtionals are permitted at execute

level.

updater(f)
A doublet that selects or updates the function that is applied when f is called in the update sense.

valof(w)
A doublet for selecting or updating the value of the variable currently associated with the identifier whose word is w.

vars x y ... z;
Syntax word - declares x,y,...,z as identifiers subject to any relevant type modifiers (macro, operation).

N while ... then ... close;
Syntax word - introduces a conditional loop. The following expression sequence is evaluated. If it is true, the imperative sequence between "then" and "close" is evaluated and the conditional expression sequence is tested again (see "until"). Conditionals are permitted at execute level.

"
Syntax word - delimits a quoted word.

%)
Syntax word - closes a partial application (see partapply, (%).

%]
Syntax word - closes a list expression (see [%).

(
Syntax word - groups items as function arguments, etc.

(%
Syntax word - forms a closure.

)
Syntax word - closes an item grouping (see ( ).

n * m
Infix operation of precedence 4 - multiplication.

n + m
Infix operation of precedence 5 - addition.

,
Syntax word - separates elements of a expression sequence (see ( ).

n - m
Infix operation of precedence 5 - subtraction.

->
Syntax word - the assignment arrow for variable assignment or the application of a function updater.

.

Syntax word - causes a function to be obeyed (see apply, ( ).

n / m

Infix operation of precedence 4 - real division.

i // j

Infix operation of precedence 4 - integer division. Returns the remainder and the quotient.

x /= y

Infix operation of precedence 7. True if x and y are not the same item, otherwise false (see equal, not).

:

Syntax word - specifies a label (see goto).

R x :: y

Infix operation of precedence 2 - infix form of conspair.

;

syntax word - terminates an imperative.

x < y

Infix operation of precedence 7. True if x is less than y, otherwise false.

l1 <> l2

Infix operation of precedence 2 - joins two lists.

x = y

Infix operation of precedence 7. True if x and y are the same item, otherwise false (see equal, /=).

n =< m

Infix operation of precedence 7. True if n is less than or equal to m, otherwise false.

=>

Syntax word - (1) at execute level, causes the items of the stack to be printed. (2) in a function declaration, separates formal parameters from formal results. (3) in a section declaration, separates shared names from exported shared names.

n > m

Infix operation of precedence 7. True if n greater than m, otherwise false.

n >= m

Infix operation of precedence 7. True if n greater or equal to m, otherwise false.

[

Syntax word - opens a list constant, closed by].

[%

Syntax word - opens a list expression, closed by %].

]

Syntax word - closes a list constant (see [ ).

n ^ m

Infix operation of precedence 3 - raises n to the power m

Popmess Facilities

The "popmess" function is used to send messages to Unix for opening and closing files, and for calling the Shell. The argument to "popmess" is a list. The head of the list is a control word, and the tail of the list is a file specification or some other argument. The following arguments to "popmess" are available:

[in filename]

Returns a character repeater function that is used to return successive characters from the specified file. When the end of the file is reached, the item named termin is returned, and the file is closed.

Example: popmess([in /us/bill/tiger]) -> c; if c()=termin then ...

[out filename]

Returns a character consumer function that is used to return successive characters from the specified file. The item named termin is consumed to mark the end of the file and to close the file.

Example: popmess("out" :: filespec) -> x; x(#a); x(termin);

[close f]

Closes the file associated with the specified character consumer or repeater.

Example: if f is the function returned by a "popmess" "in" or "out", then the file associated with f can be closed by popmess([%"close" f%]);

[shell s]

Calls the Unix Shell to execute the specified command line. The command line must be a legal POP-2 string, and must not contain a character with ASCII code 0.

Examples:

```
popmess([shell 'mv /u1/tiger mouse']);
popmess([shell 'grep "w?a*" vocab | sort +3 -nr >words']);
```

The first example renames a file. The second example creates a file consisting of all the lines of vocab satisfying the grep pattern and sorted in descending order on the third numeric field.

[exit]

A program may cause an exit from POP-2 by executing popmess([exit]).

Libraries available in Unix POP-2

The following list of libraries is available at the time of printing. Documentation for each library package is available by appending ".doc" to the library specification. For example, documentation for [lib arrays] is available in the file named [lib arrays.doc], which can be used as an argument to the standard function "poppip". Some packages may have a short tutorial introduction. If so, it will be in the file specified by appending ".intro" to the library specification, such as in [lib traps.intro].

[lib arrays]
Compiling [lib arrays] makes available the standard array functions boundslist, newanyarray, and newarray.

[lib breaks]
Provides definitions for popbreak and errfun so that the state of computation at the time of errors and ^c interrupts can be examined, modified, and suspended.

[lib filein]
Provides functions for remembering the origin of compiled functions and traces them as they are compiled.

[lib format]
Provides formatted output under control of a format specification string.

[lib fred]
Uses [lib filein] to provide a function remembering editor interface.

[lib records]
Provides prettier syntax for declaration of record classes.

[lib strings]
Provides a few useful string processing functions.

[lib text]
Provides prettier syntax for specifying the results of macros.

[lib traps]
Provides debugging aids such as setting traps for function entry and exit. There is a [lib traps.intro] file.