INDEX

1st set of directories

1) Various group information files
   Documentation, eg: (km 160,+++)
   READ.ME
   EXE.DOC
   MISC.DOC etc.

2) PATH (logical name) definitions
   PATH.HLP
   ~.PTH etc
   PATHS.CLM
   SWITCH.HLP examples

3) Program maintenance
   MAKSYS

4) XREF - cross reference

5) Prolog documentation
   PROLOG.CNL
   PROLOG.NEW

6) PRESS.BIB

7) Prolog bibliography
   (see Lawrence)

8) Special SCRIBE formats

9) CD - documentation code

10)
<table>
<thead>
<tr>
<th>Location</th>
<th>Name</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>[400,405]</td>
<td>Alan Bundy</td>
<td>1500 blocks</td>
</tr>
<tr>
<td>[400,421]</td>
<td>Programs</td>
<td>2000 blocks</td>
</tr>
<tr>
<td>[400,422]</td>
<td>Richard O'Keefe</td>
<td>1500 blocks</td>
</tr>
<tr>
<td>[400,431]</td>
<td>Ray Herries</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[400,434]</td>
<td>Chris Mellish</td>
<td>1500 blocks</td>
</tr>
<tr>
<td>[400,440]</td>
<td>Jane Hesketh</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[400,441]</td>
<td>Lawrence Byrd</td>
<td>1500 blocks</td>
</tr>
<tr>
<td>[400,442]</td>
<td>0 blocks</td>
<td>(Infin)</td>
</tr>
<tr>
<td>[400,443]</td>
<td>Bill Clocksin/Rob Milne</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[400,444]</td>
<td>Library</td>
<td>5000 blocks</td>
</tr>
<tr>
<td>[400,445]</td>
<td>Visitor</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[400,446]</td>
<td>Mary-Angela</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[400,3720]</td>
<td>Impress</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[400,4321]</td>
<td>Leon Sterling</td>
<td>1500 blocks</td>
</tr>
<tr>
<td>[400,4322]</td>
<td>Bernard Silver</td>
<td>1500 blocks</td>
</tr>
<tr>
<td>[400,4323]</td>
<td>Press</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[400,4324]</td>
<td>Lincoln Wallen</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[400,4325]</td>
<td>Luis Jenkins</td>
<td>2500 blocks</td>
</tr>
<tr>
<td>[400,4326]</td>
<td>Dave Flummer</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[400,4327]</td>
<td>Maarten van Sommeren</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[401,401]</td>
<td>Visitor</td>
<td>1000 blocks</td>
</tr>
<tr>
<td>[401,402]</td>
<td>Visitor2</td>
<td>1000 blocks</td>
</tr>
</tbody>
</table>
Useful Telephone numbers (mainly DEC10 oriented)

<table>
<thead>
<tr>
<th>Department</th>
<th>Phone Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine room</td>
<td>70-2904</td>
</tr>
<tr>
<td>ERCC Comms</td>
<td>70-2737</td>
</tr>
<tr>
<td>Ansaphone</td>
<td>9-668 2547</td>
</tr>
<tr>
<td>Keith Farvis</td>
<td>70-2661</td>
</tr>
<tr>
<td>Dave Mercer/John Payne</td>
<td>70-2627</td>
</tr>
<tr>
<td>Jeff Phillips/Roger Hare</td>
<td>70-2619</td>
</tr>
<tr>
<td>Janet Dalitz</td>
<td>70-2925</td>
</tr>
<tr>
<td>Charles Mackinder</td>
<td>70-2926</td>
</tr>
<tr>
<td>Callum Youns</td>
<td>70-2617</td>
</tr>
<tr>
<td>Frank Barratt</td>
<td>70-2604</td>
</tr>
<tr>
<td>HFS terminal room</td>
<td>6445</td>
</tr>
<tr>
<td>HFS node</td>
<td>6443</td>
</tr>
<tr>
<td>Lawrence Byrd</td>
<td>6831</td>
</tr>
<tr>
<td>FH terminal room</td>
<td>2298</td>
</tr>
<tr>
<td>Robert Rae</td>
<td>2355</td>
</tr>
<tr>
<td>CS terminal room</td>
<td>70-2790</td>
</tr>
<tr>
<td>KB terminal room</td>
<td>70-2801</td>
</tr>
<tr>
<td>CAAD</td>
<td>4566</td>
</tr>
</tbody>
</table>
A list of documentation etc that is available at HPS that we find essential.

(NB: It is important that all the following be as up to date as possible)

Dictionary
Wall hangings
  ASCII character codes (in decimal, octal, binary and ascii)
  Prolog evaluable predicate list
  Useful Telephone numbers

TOPS10 Commands manual
TOPS10 Utilities manual
TOPS10 Monitor Calls manual
LINK10 manual
MACRO manual
TECO manual
System Reference manual

Complete Edinburgh Installation manual
Network documentation for SRCNET, UC-LONDON, ARPANET

Prolog manuals
POP2 book and current guide
LISP manuals (Rutgers etc)

FINE manual
TEC124 manual
ECCE manual

SCRIBE manuals
Listings of all Scribe database files (.DEV,.MAK,.LIB,.TFO,.REF)

BLISS 10 manual
SIMULA manuals
IMP manuals
MIC documentation
COJOB documentation
FMS documentation
SUBFIL documentation
TELL documentation
BACKUP documentation
MAKLIB documentation
DDT documentation
File Daemon documentation

CIFER manual
[400,444] is the Mecho/Press project library area. This file provides pointers to other files for more information on the various facilities it provides.

If you intend to use this library, and you are not already known to us, then please send a message to [400,441] (Lawrence Byrd) so that you can be put on the mailing list for changes etc.

EXE.DOC[400,444] describes all the programs available.

MIC.DOC[400,444] describes all the mic macros available.

PATHS.HLP[400,444] describes how to initialise path definitions.

JUNK.DOC[400,444] describes a few other useful files.

MACROS.DOC[400,444,TEC] and EI.DOC[400,444,TEC] describe Lawrence's macros etc for the TEC124 editor (a better version of teco). All the 'ei' macros are kept in the directory [400,444,TEC].

PROLOG.NEW[400,444] describes all the new features present in the latest version of DEC10 Prolos held in [400,444].

UTIL.HLP[400,444,UTIL] lists all the Prolos utilities available in the program UTIL. The sources of all the UTIL modules are kept in the directory [400,444,UTIL].

MAKSYS[400,444] provides a simple program database for Mecho/Press project Prolos systems; which will allow them to be automatically reloaded when the Prolos system is changed.
This file describes all the EXE programs that live in the Mecho library area ([400,444]). Given Magic Bits and the appropriate SYS$ PATH definition these will all appear as commands to you. ALL the documentation referred to below, and that includes this file itself, can be found in the Hope Park Square terminal room (so don’t print your own copies unless really necessary). Some of these programs are just links to other (often system) programs with perhaps some CCL entry preparation thrown in. The MACRO sources for these are kept in [400,444,MAC]. These links are usually only a page long and are easy to copy for links to other things if needed.

---

**General utilities**

**CD <Place>**

Change directory to some path.

This is now a proper monitor command with the exe file in SYS$.

The documentation is in HLP: (and DOC:), just type "help cd".

**TAN**

This cleans up your area by deleting various files for you.

It currently tries to delete: *.BAK[-], PROLOG,LOG[+],

*.FIN[+] and *.BAK[+]. If it deletes the bak files from your current path and cleans up your home directory as well.

**EDX**

This is Lawrence’s POP2 editor + goodies. One day it will be a nice version of DOPE, but currently it is an undocumented collection of rubbish.

**F <file>**

Run FINE on a file. F has the nice features of adding a period to the file name if needed, and remembering the last file F’ed so that "F" without a file name uses that last file.

**QQ**

Run QUOLST (quick convenience)

**SETPATH**

Set up path definitions. This calls PATH and tells it to use "PATHS,ccl" to define path logical names. It simulates the action that normally occurs with our login switches (see PATHS,HLP[400,444] for details). It can be used at any time to update your path definitions, for example: if you have changed your PATHS,ccl. Notice that it will use the first "PATHS,ccl" it finds, which may not be your own one if you have CD’d to some other place. It is usually a good idea to CD home before using SETPATH. Alternatively it is sometimes useful to CD to someone else’s directory and use SETPATH if you want to set yourself up with their standard paths.

Note that SETPATH is always incremental - unless redefined, old paths won’t go away.

**T**

Run TEC124 (quick convenience).

See MACROS,DOC[400,444,TEC] if you are a serious tec124 user.

**TALK**

A program which allows cross talk between several users. This is more convenient than doing SENDs if you have a lot to say - especially if there are more than two people involved. Just type "TALK" - it is self documenting. All users currently running the program will share a conversation; anything anybody types appears on all the other terminals. It is often useful to do *.send 66 type ‘run mecitalk’ * at people who
you are SENDings with if you would prefer TALK but they don’t know about it (66 is arbitrary in this example - substitute their actual tty number).

XREF
Run PLL:XREF (quick convenience).
This is the Prolos Cross Referencer. Documentation is in PLL:XREF.HLP.

Language Systems
----------------
MUTIL
Minimal UTIL. This is the Prolos system plus a subset of the Mecho project utilities package. It doesn’t include: READIN, MULTIL, LONG and TIDY.

UTIL
Utilities package. This is the Prolos system plus all of the Mecho project utilities package, which includes all sorts of things - including, notably, the rational arithmetic package (LONG). Documentation is somewhat lacking but the file UTIL.HLP lists all of the procedures both by module and alphabetically. The directory [400,444,UTIL] contains all the source files which you can look at. Most routines are only a couple of lines long anyway and are easy to understand.

PROLOG
The current version of DEC10 Prolos. This is kept in [400,444] to give us control of when it changes.

PLCOMP
This is the overlay for the Prolos compiler. You NEVER need to run it directly.

Mecho/Press project systems
-----------------
INTERP
Semantic Interpreter.
(Converts parse-trees to assertions).

MECHO
Problem Solver.
(Solves mechanics problems).

ESS
Press Algebra System.
(Solves equations etc).

ROB
Deterministic Parser.
(Parses natural language sentences).
This file describes all the MIC macros that live in the Mecho library area ([400,444]). Providing [400,444] in your library search list all these can just be called as "/<name>" in the usual way.

 ALL Extends search list by adding DSKC: and DSKE:. Useful when trying to access files from areas not on your normal disk (ie DSKA:). "/all none" will contract the search list by removing these devices.

 C <command> Spawn cojob to perform command. A cojob is set up, with a little initialisation (probably buggy at the moment) and then the command is executed. The command can be anything but if it is complicated then you should surround it with angle brackets ("<" and ">") otherwise mic will not pass it as a single argument to /C. A log will be made in COJOB.LOG.

 E <file> Run FINE on the file. This has been superceded by the F program (see EXE.DOC[400,444]) but may be useful for PPNs without Magic Bits.

 EDIT Run Aaron Sloman's version of the POP2 editor.

 F Run Alan Mycroft's FRIEND program. This lists the users of the system in a pretty way excluding system jobs and other junk that SY tends to show. You can also add a line in your switch.ini file limiting what PPNs are shown (see SWITCH.INI [400,441] for an example).

 LIMIT <n> Set your virtual memory limit to nK words. It is unclear whether this is a useful thing to do on the new machine (KL). If no argument is given (ie "/limit") then your current limit is just reported.

 NEW Do a DIR listing of all your new files. This runs DIR so that it shows all the files in your area, or any SFD in your area, which are on DSKA: (only) and have been created since yesterday.

 P Call PATH so that it lists all your current path (logical name) definitions. (See also PATHS.HLP[400,444] for more details on initialising these).

 PLIB Produce an alphabetical listing of the Mecho Predicate Library. This uses PLIB.FL as a list of Predicate Library files to be searched. If you have your own version this will be used instead of the standard one in PLIB:. The result will appear in the file PLIB in your current path.

 S <file> Spawn a cojob to SCRIBE a file. A cojob is set up and initialised and then SCRIBE is run on the file specified. Note that the initialisation involves moving SCRAC: to the front of your search list so that any new files will get created on SCRAC: and not DSKA: (however if a file, the .AUX or .MEM for example, already exists on DSKA: then the new version will end up on DSKA: as well — overwriting the old). A .LOG file will be produced and its name is based on the file name given to /S.
Copy the file from DSKA: to SCRA: and delete the version on DSKA:. A file specification with wild cards can also be used (but not a list).

Short FINE edit. This calls FINE on a file but restricts the window size to 4 lines before reading in the file. You may find this useful for very quick edits where you don’t want the whole screen set up (especially if you intend to search into the file to do the fix, which would redisplay the whole screen again).

Sort a file line by line into alphabetical order. This calls CSORT (see EIM documentation) with appropriate switches. It is currently rather restricted: the file should not have an extension, the output file then has extension .SRT, and only the first 10 characters are considered for the sort. I may think about improving this if people want something better.

Call DIR so that it just shows the (allocated) space used by files in your UFDs, and contained SFDs, on DSKA:. Also: */SPACE ppp* shows the space for some other programmer number in your project, and */SPACE ppp,DSKc:* uses the specified device (DSKC: in example) instead of DSKA:. Thus */SPACE ,all:* can be used to check the space occupied by your files on all devices.
Other spurious files in [400,444]

ECCE.CMD

Initialisation for ECCE. Defines the ECCE macros as follows:
  x = Show window around current line (6 up, 6 down)
  y = Show big window below current line (21 down)
  z = Show last 6 lines of file and do a GO

Windows are shown with the current line displayed as
"**HERE**" (this string should not occur in the file!). The
initialisation also switches on %f (full verification) and
prints the first line of the file (which you will see before
the edit starts).

SAVE.CCL

For BACKUP: assuming your tape has been mounted as MT: then
"@save" should be used as your first command to backup to
prepare the tape for saving stuff. (Interchange mode is set
and the tape is wound to EOT).

STORE.CCL

For BACKUP: like save.ccl - "@restore" will prepare the tape
for restoring (tape is rewound to beginning).
Documentation for Lawrence's TEC124 macros

TEC124 is a very much better version of the editor TECO. Information on it can be found as follows:

First read 'TECO Programmer's Reference Manual' (Digital)
This describes the original version of TECO.

Then look at DOC:TEC124.RND (needs runoff'ing)
This describes all the changes between the old TECO and TEC124.

There is also a help file to be got by typing 'HELP TEC124'.

The only problem with using TEC124 (or TECO) as an editor is its curious notion of "Pages". This can be set over (and any commands to do with pages ored), by always loading the complete file into the buffer. There is a macro provided which does this (MA) and this should be used at the beginnings of every edit.

BEGIN

This file explains my system of TEC124 macros. You can obtain access to my macros by taking a copy of the file DSKA:TECO.INI[400, 444] and placing it in your login directory.
This will produce the following effects whenever you run TEC124:

Various Q registers will be initialised with macros (see below). Your default area for EI (file) macros will become [400, 444, TEC].
The ES flag is set so that after searches the line is printed with a '*' marking the pointer.
A message '[TECO.INI loaded]' will be printed.

Macros available fall into two groups:

1) The Q register macros which are run by calling the initialised registers. These are described below and a list can be printed out within TEC124 by calling Q register 'H' (MH).

2) The EI macros which are stored in files in [400, 444, TEC] and are run by typing 'ei <name>**'. A list of all those currently available can be printed by typing 'ei help**', and this also describes the function of each of them. The file which is typed in this case can be found in the file EI.DOC[400, 444, TEC].

Q register Macros

(Miscellaneous)

MH - Help. Prints list of macros,
MA - Append. Appends all the rest of the file into the buffer.
It also makes sure that form feeds are retained.

ME - End of line, Moves pointer to the end of the current line.
MP - Print. Types out the current line with a "~" to mark the pointer.

(Block Moves)

Q registers 1, 2 and 3 are used for this. Q1 and Q2 are used to mark a block of text in the buffer (Q1 < Q2). (Note: "u1" places the value of ".", which is the current position, into Q1. This is the number of characters from the beginning, so watch out if you insert/delete more text!). Q3 is used to store picked up text.

MG - Grab. Place the text marked by Q1 and Q2 into Q3. This overwrites the contents of Q3 but leaves the text in the buffer.
MS - Suck. Do a Grab but then delete the text from the buffer as well.
MK - Kollected. Like a Suck except that Q3 is not overwritten, rather the text ends up at the end of what was there already.
MJ - Jerk. Like a Kollected except that the text ends up at the front of what was already in Q3.
ML - Drop. Drop the contents of Q3 into the buffer at the current position. The pointer ends up at the end of this text. The contents of Q3 are not affected.

(Command Editing)

Q registers 8, 9 and 0 are used for this. Q8 and Q9 are used to store the original pointer position and end of buffer. Q0 is used to store the command being edited. The idea is that you can place your last command into Q0 using *0, and then use the following macros to place it in the buffer so you can edit it, and then put it back into Q0 so it can be executed (MO). The command (Q0) is placed right at the end of the buffer, however the contents of the buffer are still there! Make sure you don’t accidently mangle it!

MT - Into command edit. Remember the current position, then go to the end of the buffer and drop Q0. Print what is dropped and leave the pointer at the beginning of this (This position, which was the old end of buffer, is marked by Q9 which you can use to get you back to the top of the command (Q9J)).
MM - Into empty command edit. Like MI except that Q0 is emptied first.
MO - Out of command edit. Place the command (Q9Z) back into Q0 and delete it from the end of the buffer. Then return to the position where you came from originally.
MT - Type. Type out the command and move the pointer back to the top of the command (Q9J).

(Stack Popping)

Q registers 4, 5, 6 and 7 are used for this. They will all contain text. Q4 is a prefix string, Q5 is the stack, Q6 and Q7 are destinations which are updated when the stack is popped. The stack in Q5 is thought of in terms of lines - each line is a single stack entry.

MF - Pop stack. The stack in Q5 is popped. Q6 is overwritten with the line from the top of the stack, and Q7 is overwritten
with a string constructed by tacking the prefix (Q4) onto the front of Q6 and an <escape> onto the end. MF will also return a value: True (-1) if the pop was successful, and False (0) if there was nothing on the stack.

The intended purpose of this macro is to allow one to go through a list of files (the filenames being placed one per line in Q5) by repeatedly calling MF and then M7 until MF fails. (Note that "MF;" will exit iterations when the stack runs out). The prefix in Q4 should be the TECO I/O command you want performed (it is initially "ER"). Since an <escape> is tacked onto the end of Q7, its text ends up as a macro which performs the operation on the file! Q6 can be used if you just want the top line of the stack (you are not munching files) or if you want to print the filename out (Q6=). There are some EI macros which use MF ("ei search**", "ei roll**"); look at these for hints.

Well - thats the end of that! You will notice that this has used up a lot of Q registers! But its not that bad: Q1 -> Q0 can be used for other things when you are not using the particular macros which use them, and you can always write Q registers you are not actually using. (But don't expect EI macros to work if you do this! However "ei teco**" can be used to reinitialise the Q registers if needed). Here is a quick visual guide to what's used up. Upper case means initialised, lower case means free.

<table>
<thead>
<tr>
<th>block moves</th>
<th>stack popping</th>
<th>command editing</th>
</tr>
</thead>
</table>

1 2 3 4 5 6 7 8 9 0

g w E r T y u I O P

A S D F G H J K L

z x c v b n M

(ei register)

EI macros

Type "ei help**" to set a list of these. Note that EI macros will often assume the presence of the Q register macros described above. (See the file EI,DOC[400,444,TEC]).

END

Bugs, suggestions and monies to:

Lawrence Byrd
Dept. of Artificial Intelligence
Hope Park Square
Edinburgh
Lawrence's TEC124 macro library.

---

NB: Many of these macros are very special purpose. This is because I knock them together as I need them, and have no time to waste generalising and cleaning them up.

It is thus also the case that they are unlikely to be particularly robust. While none of them cause me any problems I can offer no guarantees as to what they will do.

ei aj$$ Display a file a page at a time. Beeps (^G) are sent to indicate the current state - one beep means another page to come, three beeps means nothing left. Type <LF> for the next page or <ESC> to abort/finish. Anything else (eg space) will redo the beeps and it is usually a good idea to do this before each page (due to buffering three beeps at the end of the previous page sound like one). At the end of the macro (after <ESC> at any stage) the buffer will be empty. I use this for printing single sheet letters etc on the AJ printer (hence the name).
(Destroys buffer).

ei bibk$$ Kollect (using MK) the SCRIBE bibliography entry surrounding the pointer. This is currently very simplistic and assumes that there are no internal @ symbols. This has turned out to be true most of the time I have used it.

ei blat$$ Go through file offering to delete lines. Type a <SPACE> to keep the line displayed or <ESC> to stop. Anything else will delete the line. The line displaying currently assumes a CIFER terminal.

ei change$$ Program for producing command files for use with the MAStape program CHANGE. Follow the instructions.
(Destroys buffer).

e. date$$ Insert today's date into the buffer at the current position. The month is inserted by name (eg "January").

ei defase$$ Depase a buffer containing form feeds into lots of little files - PAGE1, PAGE2 etc.
(Destroys buffer).

ei eb$$ Ask the user for a file name (using ei strings$) and EB it. The old contents of the buffer are flushed. The file name provided is remembered in Q6 and the EB command itself is remembered in Q7. This macro was written to provide an easy way of setting up Q6 with the current file name so that things like ei fine$ could be used easily.
(Destroys buffer).

ei f$$ Do an EX and jump to the FINE editor.
(Exits tec124)

ei fine$$ Write out the buffer and jump to the FINE editor with the file whose name is in Q6. (This is done by setting up a TMP
file for FINE and using CCL entry). Q6 can be set up either by hand, or from stack popping, or by using "ei eb$" (above). (Exits tec124)

ei fls$$
Convert a file produced by "dir /i" into a CCL file suitable for use by PRINT. Prompts for the file name (using ei eb$ and thus ei string$), does the transformation and exits with EX. Note that the path parts etc are stripped out (by ei nice$). (Exits tec124)

ei help$$
Print this documentation!

ei lc$$
Lower case from pointer to end of buffer.

ei lines$$
Give the number of lines in the buffer (has bugs/funnies - I keep meaning to rewrite this...)

ei nice$$
Clean up a buffer, which is assumed to contain a file produced by a "dir /i". All directory references are removed and the file is lower cased (ei lc$). The pointer is left at the top.

& prolog$$
The latest version of TECO Prolog. The current version is compatible, in some respects, with Prolog-10 version 3. (Destroys buffer)

ei qf$$
Query Replace. Prompts for old and new strings and does a FINE stule Query Replace through the buffer (from pointer). Terminate the strings with <ESC>, ^G at any stage will abort. Type "?" for query options.

ei roll$$
Start rolling through a list of files to be edited. Q5 should contain a list of filenames (one per line). The Q register N is loaded with a macro which will, write out the last file and (fully) read in the next one using ER. MN therefore rolls you through the files, ei roll$$ will have rolled in the first one to start with. (Destroys buffer).

ei rthru$$
Apply Q0 to all the files in Q5 (one file name per line). This is like ei thru$ except that files are ER'd, Q0 is performed, and the buffer flushed. Ie Nothing is written back to the files.

ei search$$
Search through a list of files for some strings, ei string$ is used to read a strings from you and this is searched for in the list of files in Q5. Files are ER'd and searched until the string is found. Recalling starts with the next file. (Destroys buffer).

ei stamp$$
Time stamp the buffer. If the first ten lines of the buffer contains the string "Updated!" (either case), then the text following this on that line will be replaced by a time stamp for the current time and day (using ei date$).

ei strings$$
Prompts for a string, reads it in from the terminal and places it in QQ. This macro handles <RUBOUT> and ^H (both of which delete the last character typed) but no other deletion handling. Type <ESC> to finish the string or ^G to blow the macro up (with a spurious error). ^G will also prevent this macro from returning to any macro which called it (es such as
some of the other ones documented here).

ei subit$

Prompts for a file name (using ei eb$ and thus ei strings$),
eats the file by doing ei nice$ and adding the file name
to the top of the file and then jumps to FINE with the file
using ei fine$. The intention here is that the monitor
command "dir /ir xxx,sub=" can be used to start building a
.SUB file, this macro cleans it up a bit and leaves you in
FINE to finish it off. I find myself doing this quite a lot.
(Exits tec124)

ei swini$

Distribute the "General items" section of SWITCH.INI[400,441]
to the other members of the Mecho project.
(Destroys buffer).

ei tabout$

Tabout out the end of a line to the tabstop given as an
argument. This macro expects a numeric argument which is
taken as a tabstop where tabstops occur every 8 characters.
Tabs are inserted at the end of the current line until the
end of the line reaches the specified tabstop. For serious
use this macro should probably be placed in a Q register
(e.g! 'eP tabout$ [*]x' to move it into QX and then, es$ 6MX
to call it).

ei teco$

Initialisation of all Lawrence's standard Q register macros.
Can be used to re-initialise if necessary.

ei thru$

Apply the macro in Q0 to all files in the list in Q5 (one file
name per line). Files are EB'd and MO is performed before
writing them back out. (see also ei rthru$).

ei tr$

Produce SSNAME listings from a full BACKUP directory listing.
Buffer should contain a file produced by the 'print' BACKUP
command. It will be transformed into a pretty resume'.
(Destroys buffer).

ei uc$

Upper case from pointer to end of buffer.

Use files from some MAS file. This prompts for a name
(using ei strings$) and looks for a file called <name>.sub.
This file is read into Q5 with certain lines truncated/deleted.
Lines containing ';' are completely deleted; lines containing
just ';' are truncated from just before the ';', and all lines
have trailing blanks removed. This allows one to use a .SUB
file as both a file for use by SUBFIL, and -using this ';;'
convention - as a file listing an interesting sequence of files
for editing macros. (I have found that the .SUB file usually
lists files you don't want considered while editing and I
mark these lines with ';;').

Type out buffer (from pointer), stop when a character is typed
on the terminal - leaving the pointer on the final line shown.
(NB It can take about a page before it notices!!)
Add the following line to your switch.ini file to initialise paths on login:

```
login /run:path /runoffset:1 /tmpfile:pth:'@paths,ccl'
```

There is a default paths,ccl file in [400,444] which will set used if you don’t have your own (providing you also have a ‘login /lib:[400,444]’ switch set in switch.ini as well so that it can be found. Note that this lib definition may then get replaced by a definition in paths,ccl etc).

There is a program in [400,444] called ’setpath’ which will simulate the action of the login switch above at any time during a session. Use this if you have updated your paths,ccl. To make sure it finds the right one (your one) you should probably CD back to your directory before calling setpath.

There is mic macro ’/p’ in [400,444] which calls path so that it prints out a list of all the paths you currently have defined.

Files you should look at in [400,444] are:

- PATHS.HLP  - this file
- PATHS.CCL  - default initialisation (basic + press.pth)
- MECHO.PTH  - indirect to this for Mecho definitions
- PRESS.PTH  - indirect to this for Press definitions
- ALL.PTH    - indirect to this for both the above

If you don’t understand what the hell is going on in these files then either study the documentation for PATH (available in one of the EIM folders in the HPS terminal room, or more generally from user support), or talk to Lawrence ([400,441]).
This is the default paths initialization for people who don't have their own paths.ccl file. See PATHS.HLP for how to set this called at login.

lib:/search = [400,444]
sys: = [400,444], dsk:[1,5], dsk:[1,4]
@press.pth[400,444] ; Default is for Press users
;; MECHO.pth[400,444]
;;
;; Standard path definitions for Mecho users
;;
util:  = [400,444,util]
mecho: = [400,441,mectop,mecho]
plib:  = [400,441,mectop,plib]
plcode:=[400,441,mectop,plcode]
p100:  = [400,441,p100]
;; Standard path definitions for Press users

util: = [400, 444, util]
Press: = [400, 421, Press]
match: = [400, 421, Press, match]
arith: = [400, 421, Press, arith]
extras: = [400, 421, Press, extras]
imPres: = [400, 4321, imPres]
homos: = [400, 4322, homos]
; Standard path definitions for both Press and Mecho users

Mecho

mecho: = [400,441,mectop,mecho]
Plib: = [400,441,mectop,Plib]
Plcode: = [400,441,mectop,Plcode]
P100: = [400,441,P100]

Press

@Press.pth
SWITCH.INI[400,441] — Lawrence’s initialisations

; Here are some example CD entries

cd/protections:=(self=005,group=005,others=000)
cd/name:=(alan=[400,405])
cd/name:=(chris=[400,434])
cd/name:=(leon=[400,4321])
cd/name:=(bernard=[400,4322])
cd/name:=(henry=[4060,4060])
cd/name:=(prolos=[400,447,prolos])
cd/name:=(phil=[140,143])
cd/name:=(pop=[140,141])
  name:=(mo=[400,405,my GLfloat,my GLfloat])
cd/name:=(co=[400,405,my GLfloat,coast])
cd/name:=(tec=[400,444,tec])
cd/name:=(mac=[400,444,mac])
tell/group:=(users=AllofUs+Bowen)

; General Items

; These entries are shared by all members of the Mecho project.
; Changes will be automatically distributed to your own switch.ini
; files (overwriting previous section) so don’t put stuff in here.
; If you would like additional entries added talk to Lawrence.

cd/name:=(doc=[400,421,doc])
cd/name:=(bib=[400,444])

; NB previous cd entries replaced by logical path names (see mec/paths.hlp)

tell/group:=(AllofUs=MechoProject+PressProject)
  l/group:=(MechoProject=Alan+Chris+Lawrence+B+Rob+Richard)
  t.1/group:=(PressProject=Alan+Leon+Bernard+Richard+Lawrence+B)
tell/group:=(Al2#Students=[400,445])
tell/group:=(Alan+Bundy=[400,405])
tell/group:=(Chris+Mellish=[400,434])
tell/group:=(Lawrence+Byrd=[400,441])
tell/group:=(Rob+Milne=[400,443])
tell/group:=(Richard+OKeefe=[400,422])
tell/group:=(Leon+Sterling=[400,4321])
tell/group:=(Bernard+Silver=[400,4322])
tell/group:=(Kay+Herries=[400,431])
tell/group:=(Henry+Thompson=Thompson)
tell/group:=(David+Warren=Warren)
tell/group:=(Fernando+Pereira=Pereira)
tell/group:=(Ferndy=Pereira)
tell/group:=(Robert+Rae=Rae)
tell/group:=(Aaron+Sloman=Sloman)
tell/group:=(Jeff+Phillips=Phillips)
tell/group:(Box=Service#Box)
tell/group:(Hassle=Payne+Mercer+Lawrence#B)
tell/group:(Hackers=Fernando#F+Chris#M+Lawrence#B+Richard+Rob#M+Henry)
tell/group:(HPSUsers=AllofUs+Fernando#F+David+Henry+[750,766])
tell/group:(ProlosUsers=AllofUs+Fernando#F+David+Bowen)
This file contains information on how to load from scratch various Prolog programs used by the Mecho and Press projects. It allows programs to be automatically reloaded when significant changes occur to the Prolog system or the UTIL library. If you wish to take advantage of this facility please add entries for your own programs along the lines of the current entries. Note that each field must be defined on a separate line with a colon after its name. The following fields are currently in use:

- **program**: The normal name of the program
- **owner**: The person nominally responsible for it
- **tell**: The TELL name of people to be informed of the load
- **path**: The path to be CD'd to before trying to load
- **load**: A DEC10 command which will perform the load — this will usually be a call to a MIC file.
- **note**: A comment (see below)

It is currently important that all the fields be defined and that they occur in the right order. Please be careful about this.

The MIC file should be available in the path and should perform all the necessary work to load the program from a fresh Prolog state and to save the core image into the suitable place. Note that if your program is normally closed up in a MAS file then the MIC macro should make sure it sets expanded and then closed up again at the end. Most of the MIC files used below follow certain conventions of mine, in particular they make strenuous attempts to catch errors, and they react to being given the argument "auto". See me if you would like a similar sort of thing built for your program (they tend to be rather hairy to write).

The following information also contains a note! field which is a comment giving some idea of where the component files are to be found. The terms "open" and "closed" mean available as individual files, and collected up in a MAS file respectively. (Eventually this information could be formalised a bit better...)

**.no and Press Project Programs**

```
$\$start\$

**Program**: UTIL
**owner**: Lawrence
**tell**: Lawrence@Byrd
**path**: [400,441,util]
**load**: /util auto
**note**: Always open

**Program**: MUTIL
**owner**: Lawrence
**tell**: Lawrence@Byrd+Leon@Sterling
**path**: [400,441,util]
**load**: /mutil auto
**note**: Always open
```

```
<table>
<thead>
<tr>
<th>Program</th>
<th>Name</th>
<th>Owner</th>
<th>Tell</th>
<th>Path</th>
<th>Load</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>MECHO</td>
<td></td>
<td>Lawrence</td>
<td>MechoProject</td>
<td>[400,441,mecto]</td>
<td>/mecho auto</td>
<td>Usually closed in mecho.mas</td>
</tr>
<tr>
<td>ROB</td>
<td></td>
<td>Rob</td>
<td>Rob#Milnet+Chris#Mellish</td>
<td>[400,443,gram]</td>
<td>/makrob auto</td>
<td>Usually most files closed in rob.mas</td>
</tr>
<tr>
<td>PRESS</td>
<td></td>
<td>Leon</td>
<td>PressProject</td>
<td>[400,421,press]</td>
<td>/Press auto</td>
<td>Currently open across many different paths</td>
</tr>
<tr>
<td>IMPRES</td>
<td></td>
<td>Leon</td>
<td>Leon#Sterling</td>
<td>[400,4321,impress]</td>
<td>/impress auto</td>
<td>Open across various paths</td>
</tr>
<tr>
<td>XREF</td>
<td></td>
<td>David</td>
<td>Bowen+Lawrence+Byrd</td>
<td>[140,143]</td>
<td>/xref auto</td>
<td>Open (with some .def files lying about)</td>
</tr>
</tbody>
</table>
; PRESS, MIC - Load Press

; This junk allows for automatic loading believe it or not

; Call as:
; /Press       - to load Press (normal use)
; /Press auto  - used by MAKSYS

:on error: back to death
:error ?
:on operator: back to death
:operator !
:soto cont
dead!!

:^C
:^C

.if ($a = "auto") , let el = "error"
! PRESS, MIC HALTED
:mic return
:cont!!

:let y = $date,["-",20], d = $date,[1,"-"]+"+$y,[1,"-"]+"+$y,["-",4]
:if ($d,[1] = "0") , let d = $d,[2,20]

:un util[400,444] '<revive>' ; Must use UTIL
:*: [filin],
:*: version('Press Algebra System ('d)
:*: Copyright (C) 1981 Dept. Artificial Intelligence, Edinburgh'),
:*: asserta( version_date('"d')),
:*: ok,
:save press[400,444]
MECHO.MIC - Load Mecho 'silence

This junk allows for automatic loading believe it or not

Call as:
/mecho - to load mecho (normal use)
/mecho auto - used by MAKSYS

.on error:backto death
.error ?
.on operator:backto death
.operator !
.sgoto cont
deadth::
*C
*C
.if ($a = "auto") .let e1 = "error"
! MECHO.MIC Halted
.mic return
cont::
.let y = $date,["-",20], d = $date,[1,"-"]+ "$y,[1,"-"]+ "$y,["-",4]
.if ($d,[1] = "0") .let d = $d,[2,20]
  f ($a # "auto") .sgoto L1
  redir 'revive
  dsksa:[400,441,mecto,.savex]
  .cd 400,441,mecto,.savex
  .subfil
  *mecho,mas
  *C
L1:;
; Use UTIL
.run util[400,444] 'revive
* :- [mecho],
* :- version(''Mecho Problem Solver (''d"
* Copyright (C) 1981 Dept. Artificial Intelligence, Edinburgh''),
* :- core_image, display(''Mecho Problem Solver (''d''), ttyn1,
*  reinitialise.
save mecho[400,444]
.if ($a # "auto") .sgoto L2
  e1 x.*[400,441,mecto,savex]
  j 400,441,mecto
.del x.savex,sfd
L2;:
; UTIL.MIC - Load Util

; <silence>

; This junk allows for automatic loadings believe it or not
;
; Call as: /util - to load util (normal use)
; /util auto - used by MAKSYS
;
; on error: backto death
; error ?
; on operator: backto death
; operator !
; goto cont
; death!!
*^C
*^C
; if ($a = "auto") , let e1 = "error"
! UTIL.MIC HALTED
; mic return
cont!!
; let y = $date,["-",20], d = $date,[1,"-"]+"$y,[1,"-"]+"$y,["-",4]
; if ($d[1] = "0") , let d = $d,[2,20]

 Use latest version of Prolog

; run prolog[400,444] <revive>
* :- [util],
* :- version("Utilities Package (d)
*/ Copyright (C) 1981 Dept. Artificial Intelligence, Edinburgh"),
* :- core_image,
* display("Utilities Package (d)"), tty1,
* display("Now includes LONG and TIDY"), tty1,
* reinitialise.
; save util[400,444]
; MUTIL.MIC - Load Mutil  
; This junk allows for automatic loadings believe it or not
; Call as: /mutil - to load mutil (normal use)
; /mutil auto - used by MAKSYS
;
.on error:backto death
.on error:
.on operator:backto death
.operator !
.soto cont
death!!
*C
*C
.if ($a = "auto") .let e1 = "error"
! MUTIL.MIC HALTED
.mic return
cont!!
.let y =$date,[-",20], d = $date,[1,"-"]+ "y,[1,"-"]+ "y,["-",4]
.if ($d,[1] = "0") .let d = $d,[2,20]
;
.run prolog[400,444]  
* :- [mutil].
* :- version( "Minimal Utilities Package ("d)
* Copyright (C) 1981 Dept. Artificial Intelligence, Edinburgh").
* :- core_image,
* display("Minimal Utilities Package ("d)
* reinitialise,
* save mutil[400,444]
This MIC file is for automatic loading of Rob's parser
(Please tell Lawrence if this is changed significantly)

Call as:  /rob         - to load rob (normal use)
         /rob auto     - used by MAKSYS

.on error: back to death
.error ?
.on operator: back to death
.operator!
.soto cont
default:
*C
*C
.if ($a = "auto") ,let el = "error"
! MAKROB.MIC HALTED
.mic return
cont::
.let y = $date,["-",20], d = $date,[1,"-"]+"yyyyMMdd",l= $y,["-",4]
    if ($d,[1] = "0") ,let d = $d,[2,20]
    ($a = "auto") ,soto L1
.credir '<revive>
dska: [400,443,sram,xmakx]
*C
.cd 400,443,sram,xmakx
.subfil
*rob,mas
*C
L1::

Use Latest Prologs

.run prolos[400,444] '<revive>
* :- [rob],
* :- version(''ROB - Deterministic Parser (''d)
* Copyright (C) 1981 Dept. Artificial Intelligence, Edinburgh''),
* :- ok.
.save rob[400,444]
.if ($a = "auto") ,soto L2
 'el *.[400,443,sram,xmakx]
     400,443,sram
    .del xmakx.sfd
L2::
function mksys;
  
  vars MICBuffer
      Prod Tell Path Load;
  ! Construct MIC file in here
  ! For field values of each entry
  QUIET;

  new_buffer() -> MICBuffer;
  it([mec:mksys,]);
  ! Initial PED buffer

  do_mksys();

  In MICBuffer do op([mksys, mic]); enddo;
  daz();
end;

! Do the transformation inside above setup

function do_mksys;
  
  vars Found;
      ja(); sf('$$start$$');

  findnext() -> Found;
  while Found do
      setfields();
      dropfields();
      findnext() -> Found;
  enddo;

  polishoff();
end;

! Find next program entry

function findnext => Result;
  
  ml(1);
  sft('程序:') -> Result;
  if Result then ml(-1) close;
end;

! Get the necessary fields
! NB order unimportant, but they must be there

function setfields;
vars Top;

  CP -> Top;
    set('Program:'); curfield() -> Pros;
    Jc(Top); set('tell:'); curfield() -> Tell;
    Jc(Top); set('path:'); curfield() -> Path;
    Jc(Top); set('load:'); curfield() -> Load;
end;

function curfield => Field;

  vars Start;
    while cc <= 32 do mc(1) enddo;
  CP -> Start;
  mel(0);
  mc(-1); while cc <= 32 do mc(-1) enddo; mc(1);
  rw(Start,CP) -> Field;
end;

! Create the right MIC file entries in the MICBuffer

function dropfields;

  In MICBuffer do
    itext(
      '.', cd @Path
      .let e = "ok"
      @Load
      .if ($e $ "error") .soto TELL
      .tell "b
      Problem while loading @Pros
      ^Z
      .soto NEXT
      TELL:; ;
      .tell @Tell
      ^Z
      Next:;
      ")
    ; )
    enddo;
end;

! Add finishing touches

function polishoff;

  In MICBuffer do
    ja();
    in(';', MAKSYS,MIC;
    .error ?
    .on error: backto death
soto cont
death::
! MAKSYS HAS DIED - BAD NEWS
mic exit
cont::
,let b = "Lawrence Byrd"
,let y = $date,[-",20], d = $date,[1,-"]+"+$y,[1,-"]+"+$y,["",4]
,if ($d,[1] = "0") ,let d = $d,[2,20]

enddo;
end;
; MAKSYS.MIC
;
.error ?
.on error:backto death
.soto cont
dead::
! MAKSYS HAS DIED - BAD NEWS
.mic exit
cont::
.let b = "Lawrence#Byrd"
.let y = $date,[-",20], d = $date,[1,-",]++$y,[1,-",]++$y,[-",4]
.if ($d,1 = "0") let d = $d,[2,20]
.cd [400,441,util]
.let e = "ok"
 utilis auto
.if ($e $ error") .soto TELL
tell 'b
Problem while loading UTIL
^Z
.soto NEXT
TELL::
"Ill Lawrence#Byrd

UTIL reloaded at `<time>` on `d`

^Z
NEXT::
.cd [400,441,util]
.let e = "ok"
.utilis auto
.if ($e "$error") .soto TELL
tell 'b
Problem while loading MUTIL
^Z
.soto NEXT
TELL::
tell Lawrence#Byrd+Leon#Sterling

MUTIL reloaded at `<time>` on `d`

^Z
.T::
.cd [400,441,mechop]
.let e = "ok"
/mechop auto
.if ($e "$error") .soto TELL
tell 'b
Problem while loading MECHO
^Z
.soto NEXT
TELL::
tell MechopProject

MECHO reloaded at `<time>` on `d`

^Z
NEXT::
.cd [400,443,gram]
.let e = "ok"
/makrob auto
.if ($e & "error") .soto TELL
.tell 'b
Problem while loading ROB
^Z
.soto NEXT
TELL:::
tell Rob#Milne+Chris#Mellish

ROB reloaded at '<time>' on 'd
^Z
NEXT:::
.cd [400,421,Press]
.let e = "ok"
/Press auto
.if ($e & "error") .soto TELL
.tell 'b
Problem while loading PRESS
^Z
.soto NEXT
TELL:::
  ^l PressProject

PRESS reloaded at '<time>' on 'd
^Z
NEXT:::
Buffer should start containing filename
Also set QZ if runoff should be called

Q1 = pointer
Q2 = pointer
Q3 = position save state (after FILE while searchins for block)
Q6 = filename
Q7 = :er/:eb constructed command
QU = result of last :er/:eb performed in an M7 command
Q8 = True/False flag indicates success of last major operation.
Q9 = Result flag (level 1)
Q0 = Result flag (level 2)
Q0 = Current declaration block
QX = rest of declaration file
QZ = Flag - whether to run runoff on exit

-1eu ! Seems to be wrong otherwise!
1<
hx6 ji:er^C zj i

[Uu]^C hx7 hk m7 au^U ^A
Cannot find Declarations file: ^A q6= 0;'
^A
XREF file updataer
^A
ma hxx
< HK sx j
sFILE"^C ;,u1 ma,u2 a1,a2x6 i


Uu"^C ,u2 a1j i:eb"^C a2+3u2 a1,a2x7

-1u8 a1u3
:sp declareations%^[U9
 a9'U 0u8'
 a9'S 01 .u1 :s%end%^[U0
 o0'U 0u8'
 o0'S 1 .u2'

a8'F ^A
** Fatal Error
Invalid Declarations file: ^A q6= ^A File Position: ^A q3= 0;'
a1,a2x8 0,a2k hxx
HK
^A
^A 0,a6= ^A ^A
-1u8
m7 au^U ^A ** read failure ^A 0u8'
au'S ma :sp declareations%^[U9
 a9'U j :s%here%^[U0


\texttt{a0'S 01 s0' u1 :s%end%\texttt{U0}}
\texttt{a0'S 01k s0' \texttt{A} ** no matching %end% \texttt{A 0u8'}}
\texttt{a0'S 1 ,u2 a1,a2k s0' \texttt{A}}

\texttt{a8'T hp :efU9}
\texttt{a9'S \texttt{A} ** write failure}

** Fatal Error - unrenameable TMP file produced accidentally.
Processing stops.
\texttt{A 0;}
\texttt{a9'S \texttt{A} OK \texttt{A'}}
\texttt{a8'F ek'}

>}

Exit

\texttt{A}
\texttt{aZ'T led sys:runoff\texttt{A} \texttt{ARunoff: A'}}
\texttt{A'}
\texttt{Z}
mode widen(?,-),
mode writepred(+,+),
mode writes(+),
mode yesno(+,-).

Data for user definitions.
The following terms may be recorded using a predicate as the key:

$system for built-in predicates.
$known(Where) for predicates known to be defined in "Where",
$appplies(P,T) for predicates P which apply one of their arguments T
$called for predicates which are called from other places

Load in definition file containing system or known predicates,
or operators

load([F,,L]) :- !, ( load(F) ; true ), !, load(L),
load([]) :- !,
load(File) :- !,
    see_chek(File),
    repeat,
    read(T),
    (T=end_of_file; note(T), fail),
    seen.

Process terms in definition file,
note(system(P)) :- !, crecord(P,’$system’),
note(known(P,Where)) :- !, crecord(P, ’$known(Where)’),
note(op(Prec,Assoc,Name)) :- !, call(op(Prec,Assoc,Name)),
note(appplies(P,T)) :- recorda(P, ’$applies(P,T)_’), !,
note(called(P)) :- mark_interpreted(P).

data
$caller(Called_Predicate, Callins_Functor, Callins_Arity, Where Defined)
$cdefn(File, Predicate)
$file(File)
Predicate

Top level and creating initial database

so :- nolog, ’LC’,
    repeat, ttyp1, display(’Next file!’), ttypflush,
    readton1(Ch),
    reply(Ch), !.

check for indirection: @<filename> %

reply([]) :- !, collect!,
reply([&4|Ch]) :- !,
    name(F,Ch), indirect(F),
    reply(Ch) :- name(F,Ch), setfrom(F).
% set file names from indirect file %

indirect(C) :- see_check(C), repeat, dofile, !, fail.

dofile :- read_atom(Ch), !, name(F, Ch),
          display('Next file: '), display(F), trynl,
          setfrom(F),
          dofile :- seen.

% Saving current input file, process Prolog code in file F %

setfrom(F) :- seeins(G), see_check(F),
            recordz('$$file'(__, '$file'(F, __)),
            repeat, exhaust(F), !,
            seen, see(G), fail.

% Process each clause, T, in file F %

exhaust(F) :- read(T), expand_term(T, T1), process(T1, F), T=(end_of_file).

process(P:I, I, _):- !, head(P, I, F, N), soal(Q, G), caller(G, F, N, I),
                   process(Q, I, G, N, I),
                   process(Q, N, I, _), !, call(G),
                   process(Q, N, I, _), !, call(G),
                   process(end_of_file, I),
                   process(P, I) :- head(P, I, _, _).

% Record the fact that P is a predicate & that it is defined in I.
% Return the principal functor of P (F) & its arity (N).

head(P, I, F, N) :- functor(P, F, N), functor(G, F, N), definition(G, I), !.

% Fail if soal uninstantiated (how can this happen?), Otherwise return most
% general term having the principal functor & arity of each soal in the
% clause (successively on backtracking). Ignore system predicates.

soal(G, _) :- var(G), !, fail,
soal(G, G1) :- soal0(G, G1).

soal0(G, _, G1) :- soal(G, G1),
soal0(_, G, G1) :- !, soal(G, G1),
soal0(G, _, G1) :- soal(G, G1),
soal0(_, G, G1) :- !, soal(G, G1),
soal0(X,F, G) :- !, soal(P, G),
soal0(else(G, _, G1)) :- soal(G, G1),
soal0(else(_, G, G1)) :- !, soal(G, G1),
soal0(then(_, G, G1)) :- !, soal(G, G1),
soal0(G->_, G1) :- soal(G, G1),
soal0(_, _->G, G1) :- !, soal(G, G1),
soal0(not(G), _, G1) :- !, soal(G, G1),
soal0(\+(G), G1) :- !, soal(G, G1),
soal0(call(G), G1) :- !, soal(G, G1), mark_interpreted(G1),
soal0(basof(_, Gs, _, _), G) :- !, soal(Gs, G), mark_interpreted(G),
soal0(setof(_, Gs, _, _), G) :- !, soal(Gs, G), mark_interpreted(G),
soal0(G, _, _) :- recorded(G, 'system', _, !), fail,
soal0(G1, G2) :- recorded(G1, 'applies'(G1, P), _),
soal0(\+, G1, G2) :- !, soal(G1, G2), mark_interpreted(G2),
soal0(G, G).

% No cut here
% Record that P is a predicate and that it is defined in file I
% definition(P,I) :- recorded(P,'$system',_), !.
% warn(P,'already defined as a system predicate'), fail.
definition(P,I) :- crecord(''$pred',P), crecord(P,'$defn'(I,P)).

% Record that P is a predicate called by F/N in file I
% caller(P,F,N,I) :- functor(P,Pf,Fn), functor(P1,Pf,Fn),
% crecord(''$pred',P1), crecord(P1,'$caller'(P1,F,N,I)).

caller(P,F,N,I) :- functor(P,F,N,I), functor(P,F,N),
crecord(''$pred',P1), crecord(P1,'$caller'(P1,F,N,I)).

% Record that P is called by the user or outside its file of
% definition, and hence must be public
mark_interpreted(P) :- caller(P,'<user>',0,undefined).

% Record term Q on key P unless already recorded
crecord(P,Q) :- recorded(P,Q,_), !.
crecordz(P,Q) :- recordz(P,Q), !.

% Increase arity of predicate by specified amount
widen(A+,_) :- var(A), !, fail.
% NB also covers variable as first arg
iden(A+Offset,A1) :- !, functor(A,F,N1), N2 is N1+Offset, functor(A1,F,N2),
iden(A,A) :- !.

% Collecting up %

% data
% $ext(File, Predicate)
% % Held on key: File.
% % $ext(I,P) means P is an external (import) of I
% $entry(File, Predicate)
% % Held on key: File.
% % $entry(I,P) means P is an entry (export) of I

collect :-
  fentries(L),
  asort(L,L1,[[]]),
  repeat, nl,
  display('Output to: '),
  ttyflush,
readfile(Chars), name(File,Chars),
  tell_check(File), !,
  output1(File,L1),
told,
  tmpcor(tell,rno,Chars),
outsymbols.

% This will run rundown

% Find entries. Search through all encountered predicates %

fentries([e(F,N,f(I,Cs)];L]) :- recorded(''$pred',F,Ptr), erase(Ptr),
factor(F,P,N),
defn_file(P,I), callers(P,I,Cs),
(L=undefined, warn(P,'not defined'))\; true),
((Cs=C[]), warn(P,'not called')\; true),
multiple_defn(P,[I]L,L1),
((nonvar(L), warn(P,'multiple defined')\; true), !,
fentries(L1),
fentries([]).

defn_file(P,I) :- sot_defn(P,I), !.
defn_file(P,undefined).
sot_defn(P,I) :- recorded(P,'$defn'(I,P),_).
sot_defn(P,I) :- recorded(P,'$known'(I),_).

multiple_defn(P,List,[e(F,N,f(I,[J]))!L,L1]) :- % Look for multiple defs
    sot_defn(P,I), notin(I,List), !,
    functor(F,P,F,N),
    multiple_defn(P,[I;List],L,L1),
    multiple_defn(P,_,L,L).

notin(X,List) :- member(X,List), !, fail.
notin(_,_).

% Return a list of all callers of the procedure P
% This procedure does not fail (returns empty list)
callers(P,I,[c(F,N):Cs]) :- recorded(P,'$caller'(P,F,N,J),Ptr),
    erase(Ptr), !,
    exts(I,J,P),
    callers(P,I,Cs),
    llers(_,_,[]).

% Record externals. P is defined in I (entry or export), used in J
% (external or import)
exts(I,I,_), !.
exts(I,undefined,P) :- !, entries(I,P),
exts(I,J,P) :- entries(I,P), crecord(J,'$ext'(J,P)).

% Record entries%
entries(undef_defined,_) :- !.
entries(I,P) :- crecord(I,'$entry'(I,P)).

% imports and exports wanted? %
ifexts :-
    yesno('List imports and exports',yes).

% build list of externals%
setexts(F,[e(G,N,f(I,_))|Exts]) :- recorded(F,'$ext'(F,P),Ptr),
    erase(Ptr), !,
    functor(P,G,N),
    defn_file(P,I),
    setexts(F,Exts),
    setexts(_,[]).

% build list of entries%
setentries(F,[e(G,N,_)|Ents]) :- recorded(F,'$entry'(F,P),Ptr),
    erase(Ptr), !,
    functor(P,G,N),
    setentries(F,Ents),
    setentries(_,[]).

% Quick sort of functor entries%
nsort([X:L],R,R0) :-
    partition(L,X,L1,L2), nsort(L2,R1,R0),
    nsort(L1,R,[X|R1]),
    nsort([],R,R).

partition([X:L],Y,[X:L1],L2) :- lte(X,Y), !,
    partition(L,Y,L1,L2).
\texttt{\textbf{partition([X|L],Y,[L1,X|L2]) :- partition(L,Y,L1,L2),}}

\texttt{\textbf{partition([],[],[],[]).}}

\texttt{\textbf{1lt(e(A1,N1,-),e(A2,N2,-)) :- 1lt2(A1,N1,A2,N2).}}

\texttt{\textbf{1lt2(A,N1,A,N2) :- !, N1=N2.}}

\texttt{\textbf{1lt2(A1,-,A2,-) :- A1 @ A2.}}

\% Output %

\texttt{\textbf{output1(File,Dbase) :- repeat, ttynl, display(‘Width: ’),}}

\texttt{\textbf{ttysflush, readtonl(Chars),}}

\texttt{\textbf{(name(Width,Chars), integer(Width), Width>50, Width<150, !;}}

\texttt{\textbf{complain([’50 < Width <150’]) ),}}

\texttt{\textbf{ttynl, display(‘Title: ’), ttyflush, readtonl0(Title),}}

\texttt{\textbf{write(’,nap ,ps 58,’), write(Width), nl,}}

\texttt{\textbf{write(’,rm ’), write(Width), nl,}}

\texttt{\textbf{write(’,lM 0 ,ts 24,38’), nl,}}

\texttt{\textbf{write(’,no flag all’), nl,}}

\texttt{\textbf{write(’,title ’), writes(Title), nl,}}

\texttt{\textbf{write(’,s 2 ,c’), nl,}}

\texttt{\textbf{write(‘****************** **************’), nl,}}

\texttt{\textbf{write(’,c’), nl,}}

\texttt{\textbf{write(‘* PROLOG CROSS REFERENCE LISTING *’), nl,}}

\texttt{\textbf{write(’,c’), nl,}}

\texttt{\textbf{write(‘****************** **************’), nl,}}

\texttt{\textbf{write(’,s 2 ,c’), nl, writes(Title), nl,}}

\texttt{\textbf{write(’,s 3 ,nf’), nl,}}

\texttt{\textbf{tab(2), write(‘PREDICATE’), tab(15), write(‘FILE’),}}

\texttt{\textbf{tab(11), write(‘CALLED BY’), nl,}}

\texttt{\textbf{write(’,f ,s 3’), nl,}}

\texttt{\textbf{write(’,lM 38’), output2(Dbase).}}

\texttt{\textbf{output2(getDbase) :- member(e(F,N,f(I,Cs)),Dbase),}}

\texttt{\textbf{front(F,N,I), back(Cs), fail,}}

\texttt{\textbf{output2(_).}}

\texttt{\textbf{front(F,N,I) :- nl,}}

\texttt{\textbf{write(’,p -38;1,1’), nl,}}

\texttt{\textbf{writepre(F,N),}}

\texttt{\textbf{put(9), write(I), put(9), !.}}

\texttt{\textbf{back([].) :- !.}}

\texttt{\textbf{back([C|(F,N)!Zs]) :- writepre(F,N),}}

\texttt{\textbf{put(32), back(Zs), !.}}

\% Produce lists of imports and exports if wanted,
\% Afterwards, call RUNOFF to process the cross reference listings

\texttt{\textbf{outslobs :-}}

\texttt{\textbf{ifexts, !, % List of imports and exports required for each file?}}

\texttt{\textbf{repeat,}}

\texttt{\textbf{display(‘Output to: ’),}}

\texttt{\textbf{ttysflush,}}

\texttt{\textbf{readtonl(Chars),}}

\texttt{\textbf{name(File,Chars),}}

\texttt{\textbf{tell_check(File), !,}}

\texttt{\textbf{decls_to_one_file,}}

\texttt{\textbf{told,}}

\texttt{\textbf{outslobs1(Chars).}}
outslobals :- !, % no extra list wanted
run('sys!runoff',1), % run runoff

% If imports and exports have been listed, do we want to
% update the declarations in the files? If so, call TECO
% before calling RUNOFF for the cross reference listings.
% The XREF-TECO macro looks in each file, replacing any existing
% declarations with new ones. The declarations block replaced is defined
% as being:
% 1) The first block delimited by lines containing
% "%declarations%" - top
% "%end%" - bottom
% 2) The first line containing "%here%"
% If neither condition is satisfied then the file is left alone.
outslobals1(FileStrings) :-
  yesno('Alter existing files',yes), !,
  % Calling TECO is hairy
  append(FileStrings,$'-1uZeixref$',BottomBit),
  append("$xx/imp$ek.au-124";'Neiteco.tec[400,444,tec]$'i',BottomBit,Horrific),
  tmpcor(tell,edt,Horrific),
  run('sys!tec124',1),
outslobals1(_):-
  run('sys!runoff',1),

decls_to_one_file :-
  recorded('file'(_),'file'(F),F), % Yes, find a file in database
  erase(F), % Erase it
  write('FILE '), write(F), nl, nl,
  outputdcirs(F), nl, nl,
  fail,
  decls_to_one_file.

% Output 'public' and 'import' declarations for a file
% Declarations start with '%declarations%' and end with '%end%'
outputdcirs(F) :-
  settexts(F,Lx), % List its imports
  setentries(F,Le), % List its exports
  aSort(Lx,Lsx,[]), % Sort import list
  aSort(Le,Lse,[]), % Sort export list
  write('%declarations%'), nl, nl,
  do_publics(Lse),
  do_imports(Lsx),
  write('%end%'), nl.
do_publics([],_):- !.
do_publics([E|[L]]) :-
  write(':- public'), nl, one_pub_decl(E), publics1(L), nl,
  one_pub_decl(e(F,N,_-)) :- put(9), put(9), write(pred(F,N)),
  publics1([E|[L]]) :- !, put(44), nl, one_pub_decl(E), publics1(L),
  publics1([]) :- put(46), nl.
do_imports([],_):- !.
do_imports(L) :-
  write('% imports:'), nl,
  imports1(L), nl.
import1([]),
import1([E|L]) :- one_imp_decl(E), import1(L),

one_imp_decl(e(F,N,f(I,_))) :-
  put(37), put(9), put(9), write('red(F,N),
  name(F,Na), length(Na,N1),
  number_length(N,N2), tab(24-N1-N2-1),
  write('from '), write(I), put(41), n1,
  number_length(N,1) :- N<10, !,
  number_length(N,2) :- N<100, !,
  number_length(N,3),

% Utilities for input/output %
write('red('<user>' ),0) :- !, write('<user>' ),
write('red(F,N) :- writeo((F,N))

readton1(Cs) :- set0(C),
  actionchar(C,Cs),
  actionchar(31,[C]) :- !, fail,
  actionchar(C,Cs) :- C=<32, !, readton1(Cs),
  actionchar(C,[C:Cs]) :- !, readton1(Cs),

readton10(Cs) :- set0(C),
  (C>==31, !, Cs=C; C>==26, !, fail;
  Cs=[C:C1], readton10(C1) ),

writes([]) :- !,
writes([C:Cs]) :- put(C), writes(Cs).

member(X,[X|_]),
member(X,[_|L]) :- member(X,L),

append([],X,X),
append([A|B],C,[A|D]) :- append(B,C,D),

>=sno(Question,Answer) :-
  ttyln1, display(Question), display('?',), ttyflush,
  readton1(Ans),
  (Ans=[Y|_], (Y:=="y"; Y:=="Y"), !, Answer=yes;
  Answer=no ),

complain(L) :-
  ttyln1, display(' X '),
  (member(Text,L), display(Text); ttyln1), fail.

% See file or complain if it doesn't exist
see_chk(File) :-
  ( nofileerrors ; fileerrors, complain([File, 'not found']) ),
  see(File), !, fileerrors,

% Open file for output or complain
tell_chk(File) :-
  ( nofileerrors ; fileerrors, complain([Can't open ', File]) ),
tell(File), !, fileerrors,
% Give a warning about a predicate

warn(P, State) :-
    display(’** WARNING: ’), display(P),
    display(’ is ’), display(State), nl,
PROLOG CROSS REFERENCE PROGRAM

Source: David Bowen and the Mecho group
Program Issued: May 1981
Documentation: May 1981

Description

XREF is a cross-reference program which produces an alphabetically ordered list of predicates, giving the file in which each is defined and a list of all the predicates which call it. If required, it also gives a list of all the imports and exports for each file, i.e., the global predicates used and defined in each file. Output is produced in an appropriate format for RUNOFF, the text processing program.

The user may extend the program by providing a definition file. This can specify system predicates (for which cross-referencing is not required), operators, known predicates (for which definitions need not be provided), and predicates which take other predicates as arguments and cause them to be called.

2. How to Use the Program

The simplest way to use the program is to type

```
run PLL:XREF
```

The prompt "Next file:" then appears and a file name may be typed in, followed by a carriage-return. The prompt is then repeated so that as many file names as desired may be entered. If a number of file names are listed on a file called F00, say, then they can all be entered at once simply by typing

```
@F00
```

When all the required files have been entered, this input phase is terminated by just typing a carriage-return by itself.

The user is subsequently prompted for an output file name, and a title and required text width for the output. He is then asked whether a lists of imports and exports are required, and should type the single letter "y" (or "yes") if they are.

The executable version of XREF, of which the use has been described, makes use of a set of definitions which may be found in PLL:XREF.DEF. These define the standard system predicates, for which cross-referencing is not required, e.g.
system(fail),
        system(length(_,_)).

indicate that fail with no arguments and length with two arguments are built-in predicates.

The user may provide his own definition file(s). To do this it is necessary to run the Prolog interpreter and to re-compile XREF. The top-level predicate load is then available to read in definition files. e.g.

    ,run prolog
    ?- compile('PLL:XREF'),
    ?- load(['MYFILE.DEF', 'PLL:XREF.DEF']),
    ?- go.

load takes either a list of file names or a single file name as its argument, so starts the cross-reference program which then runs as described above.

Apart from the definition of system predicates, three other kinds of terms are allowed in a definition file. These are of the form

    op(<Priority>, <associativity>, <operator symbol list>),
    known(<predicate>, <where defined>),
    applies(<term>, <term>),

The op terms simply declare operators as in the normal way except that they take the form of assertions rather than goals (the preceding ':' is omitted). It is not necessary to declare operators in the definition file unless their use precedes their declaration in the files to be processed. known avoids the need for reading in the full definitions of standard predicates. It simply tells the program where a particular predicate is defined. applies can be used to indicate that a given predicate takes some other predicate as an argument and causes that predicate to be called. For example, suppose there is a user-defined predicate maplist which takes a predicate and two lists as arguments e.g.

    maplist(foo,L1,L2)

and produces a list of which the i-th element is the result of applying the predicate to the i-th elements of each of the lists. The appropriate entry in the definition file is

    applies( maplist(Pred,L1,L2), Pred+2 ).

This expresses the fact that the first argument of maplist is a predicate and that it is to be called with two additional arguments, i.e. the above call of maplist(3) results in calls foo(2). The '+2' would be omitted if maplist caused foo to be called with no additional arguments, i.e. if foo(0) was to be referenced. Note that if maplist is called with a variable as its first argument, the cross-reference cannot determine what predicate(s) are to be called and so no special action is taken.

3. Storage Requirements

The Prolog system requires 30K words. Another 15K words suffice to run the compiled version of XREF on a small program.
/* XREF.DEF - system definitions for use with XREF.PL.
   
   Updated: 20 May 81
   
   These are the functors that have special significance as predicates
   */

system(_,_,_).
system(abolish(_,_,_)).
system(revive(_,_,_)).
system(incore(_,_)).
system(asserta(_,_,_)).
system(asserta(_)).
system(assertz(_,_,_)).
system(assertz(_)).
system(retract(_,_)).
system(clause(_,_,_,_)).
system(clause(_,_,_)).
system(recorda(_,_,_,_)).
   \.
system(recordz(_,_,_,_)).
system(recorded(_,_,_,_)).
system(instance(_,_,_)).
system(erase(_)).
system(true(_)).
system(length(_,_,_)).
system(name(_,_,_)).
system(op(_,_,_,_)).
system(var(_)).
system(atom(_)).
system(!(_)).
system(statistics(_)).
system(statistics(_,_,_)).
system(functor(_,_,_,_)).
system(call(_)).
system(expand_term(_,_,_)).
system(debug(_)).
system(debug_only(_)).
system(display(_)).
   \.
system(set(_)).
system(set0(_)).
system(leash(_)).
system(nl(_)).
system(nodebug(_)).
system(print(_)).
system(put(_)).
system(skip(_)).
system(tab(_)).
system(trace(_)).
system(ttyflush(_)).
system(ttyset(_)).
system(ttyset0(_)).
system(ttynl(_)).
system(ttyput(_)).
system(ttye(_)).
system(ttytab(_)).
system(write(_)).
   \.
system(writea(_)).
system(ancestors(_)).
system(depth(_)).
% not an evaluable predicate

% not an evaluable predicate
/* UTIL.DEF : XREF definitions for UTIL procedures

Lawrence
Updated: 14 June 81

*/

% UTIL operators

op( 1100,xfy,\(\\) ),
op(  950,xfy,#  ),
op(  850,xfy,&  ),
op(  710,xfy,\[\text{not,thnot}\] ),
op(  700,xfy,\:\=  ),
op(  300,xfy,edit   ),
op(  300,xfy,redo   ),
op(  300,xfy,tlim   ),
op(  300,xfy,ton    ),
op(  300,xfy,toff   ).

% UTIL procedures

known(    \&\((\text{Goal1,Goal2})\),
         \&\((\text{Goal1,Goal2})\), \text{Goal1} ),
         \&\((\text{Goal1,Goal2})\), \text{Goal2} ),
known( \:\=\((\text{X,Y})\),
known(    \:\\\((\text{Goal1,Goal2})\),
         \:\\\((\text{Goal1,Goal2})\), \text{Goal1} ),
         \:\\\((\text{Goal1,Goal2})\), \text{Goal2} ),
known(    \text{any}(\text{Goal1list})
         \text{any}(\text{Goal1list})
\),
% Hairy applies...

known(    \text{append}(\text{List1},\text{List2},\text{List3})
known(    \text{apply}(\text{Pred},\text{Args})
known(    \text{binding}(\text{N},\text{Goal})
known(    \text{casserta}(\text{X})
known(    \text{cassert}(\text{X})
\text{own}(    \text{csensym}(\text{Prefix},\text{PossVar})
known(    \text{check\_exists}(\text{File})
known(    \text{checkand}(\text{Pred},\text{ConJ})
\text{applies}(\text{checkand}(\text{Pred},\text{ConJ}), \text{Pred+1} )
known(    \text{checklist}(\text{Pred},\text{List})
\text{applies}(\text{checklist}(\text{Pred},\text{List}), \text{Pred+1} )
known(    \text{clean}   
known(    \text{close}(\text{File},\text{Old})
known(    \text{concat}(\text{Atom1},\text{Atom2},\text{Atom3})
known(    \text{continue}   
known(    \text{convlist}(\text{Pred},\text{List1},\text{List2})
\text{applies}(\text{convlist}(\text{Pred},\text{List1},\text{List2}), \text{Pred+2} )
known(    \text{delete}(\text{File})
known(    \text{diff}(\text{X,Y})
known(    \text{disjoint}(\text{List})
known(    \text{edit}(\text{File})
known(    \text{error}(\text{Format},\text{List},\text{Action})
\text{applies}(\text{error}(\text{Format},\text{List},\text{Action}), \text{Action} )
known(    \text{file\_exists}(\text{File})
known(    \text{findall}(\text{Var},\text{Goal},\text{List})

utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
utility )
applies( forall(Var,Goal,List), Goal ),
known( flag(Flag,Old,New), utility ),
known( for(N,Goal), utility ),
applies( for(N,Goal), Goal ),
known( forall(Goal1,Goal2), utility ),
applies( forall(Goal1,Goal2), Goal1 ),
applies( forall(Goal1,Goal2), Goal2 ),
known( scc(Goal), utility ),
applies( scc(Goal), Goal ),
known( sensym(Prefix,Var), utility ),
known( intersect(Set1,Set2,ISet), utility ),
known( list(Effect,Set), utility ),
known( llistoset(List,Set), utility ),
known( mapand(Pred,ConJ1,ConJ2), utility ),
applies( mapand(Pred,ConJ1,ConJ2), Pred+2 ),
known( maplist(Pred,List1,Set2), utility ),
applies( maplist(Pred,List1,Set2), Pred+2 ),
known( member(Effect,Set), utility ),
known( memberchk(Effect,Set), utility ),
known( mlist(Pred,Lists), utility ),
applies( mlist(Pred,Lists), Pred+1 ),
known( mlmaplist(Pred,Lists), utility ),
applies( mlmaplist(Pred,Lists,Win,Vout), Pred+3 ),
known( mlmaplist(Pred,Lists,V), utility ),
applies( mlmaplist(Pred,Lists,V), Pred+2 ),
known( mllmember(Elements,Lists), utility ),
known( mllselect(Elements,Lists,Rests), utility ),
known( nextto(X,Y,List), utility ),
known( rmember(Effect,Set,N), utility ),
known( not(Goal), utility ),
applies( not(Goal), Goal ),
known( not(Goal), utility ),
applies( not(Goal), Goal ),
known( numlist(N1,N2,Numberlist), utility ),
known( occ(X,Term,N), utility ),
known( open(File), utility ),
known( open(Old,File), utility ),
known( pairfrom(List,A,B,Rest), utility ),
known( perm(List1,List2), utility ),
known( perm2(X,Y,A,B), utility ),
known( prconj(ConJ), utility ),
known( prerepr(Expr), utility ),
known( prlist(List), utility ),
known( read_in(Sentence), utility ),
known( redo(File), utility ),
known( remove_dups(List,Set), utility ),
known( rev(List1,List2), utility ),
known( select(Element,Set,Rest), utility ),
known( seteq(Set1,Set2), utility ),
known( some(Pred,List), utility ),
applies( some(Pred,List), Pred+1 ),
known( subgoal(exact,Goal), utility ),
known( sublist(Pred,List1,List2), utility ),
applies( sublist(Pred,List1,List2), Pred+1 ),
known( subset(Subset,Superset), utility ),
known( subst(Substitution,Old,New), utility ),
known( subtract(Set1,Set2,Subset), utility ),
known( sumlist(NumberList,Sum), utility ),
known( thnot(Goal), utility ),
applies( thnot(Goal), Goal ),
known(  tlim(Tlimit)),
known(  ton(Name)),
known(  toff),
known(  toff(Name)),
known(  trace(Format,Condition)),
known(  trace(Format,List,Condition)),
known(  ttyprint(X)),
known(  union(List1,List2,USet)),
known(  variables(Term,VarSet)),
known(  writef(Format)),
known(  writef(Format,List)),

utility ),
/* XREF.DEF - system definitions for use with XREF.PL.

Updated: 20 May 81

These are the functors that have special significance as predicates */

system([_!_]),
system(abolish(_,_)),
system(revive(_,_)),
system(incore(_)),
system(assert(_,_)),
system(assert(_)),
system(assertz(_,_)),
system(assertz(_)),
system(retract(_)),
system(clause(_,_,_)),
system(clause(_,_)),
system(record(_,_,_)),
system(record(_,_)),
system(recordd(_,_,_)),
sem(recorded(_,_,_)),
system(instance(_,_)),
system(erase(_)),
system(true),
system(length(_,_)),
system(name(_,_)),
system(op(_,_,_,_)),
system(var(_)),
system(atom(_)),
system(!),
system(statistics),
system(statistics(_,_,_)),
system(functor(_,_,_),
system(call(_)),
system(expand_term(_,_,_)),
system(debug),
system(debugins),
system(display(_)),
system(set(_)),
system(seto(_)),
system(leash(_)),
system(nl),
system(nodebug),
system(print(_)),
system(put(_)),
system(skip(_)),
system(tab(_)),
system(trace),
system(ttyflush),
system(ttyset(_)),
system(ttyset0(_)),
system(ttynl),
system(ttyput(_)),
system(ttyskip(_)),
system(ttytab(_)),
system(write(_)),
system(writea(_)),
system(ancestors(_)),
system(depth(_)),
This file lists the changes/fixes made in versions of DEC10 Prolog. The most recent version (currently) always lives in the Mecho Library area [400,444]. The system version is apt to be a somewhat earlier version. If you intend to use the most recent version then please TELL [400,441] so as to set on the mailings list since the version in [400,444] can suddenly change without prior warning. For this reason you may prefer to use the system version (in SYS:) since it will change less often, and always with plenty of warning.

DEC10 Prolog version 3.35 in [400,444]
10 August 81

This history starts here.
The documentation for this version is scattered across the following files:

- DOC:PROLOG.DOC (Original documentation)
- DSKA:GUIDE3.MEM[400,447]
- DSKA:DEBUG.MEM[400,447] (Version 3 documentation)
- DSKA:PROLOG.CNG[400,444] (List of most recent changes - this file)
- DSKA:PROLOG.NEW[400,444] (Documentation for recent changes)

One day the documentation will be rewritten and all this information gathered together.
New and Changed Features of DEC-10/20 Prolog version 3.35
=================================================================

NB: See DSKit PROLOG.CNG for a history of changes.

1. Strategy for Global Stack Garbage Collection

The evaluable predicate ‘scsGuide’ has been redefined. It has now the form

    scsGuide(Function,Old,New)

Function can be ‘margins’, which is the number of free pages below which
garbage collection is always tried, or ‘cost’, which is the percentage
of runtime the user accepts to be consumed on garbage collections. The
‘margins’ value is initially 50 pages; the ‘cost’ 10 percent. When the
procedure is called, the current value corresponding to Function is
unified with Old, and New is stored as the new value. If New is not an
integer or it is out of range, the call fails.

2. Undefined Predicates

The interpreter can optionally catch calls to predicates that have no
clauses. The state of the catchings facility is can be :-

* ‘trace’, which causes calls to predicates with no clauses to be
  reported and the debugging system to be entered at the earliest
  opportunity;

* ‘fail’, which causes calls to evaluable predicates to fail just as
  before (the default state).

The evaluable predicate

    unknown(OldState,NewState)

unifies OldState with the current state and sets the state to NewState.
It fails if the arguments are not appropriate. The (old) evaluable predicate
‘debugging’ will now print the value of this state along with its other
information. Please note that there
is a time (NOT space) overhead of about 70 % when running interpreted
programs with the facility enabled (‘trace’ state). It is hoped that this
facility will be improved in the future.

3. Two Argument Save

The evaluable predicate

    save(File,Return)

saves the current system state in File just as ‘save(File)’, but in
addition unifies Return to 0 or 1 depending on whether the return from
the call occurs in the original incarnation of the state or through a
call 'restore(File)' (respectively).

4. Running Other Programs

The evaluable predicate

run(Program, Offset)

runs the program in file Program, starting at offset Offset. Program should be an atom and Offset an integer (eg run('sysidirect', 0)). The usual Prolog restrictions on file names apply to Program (ie PPN's or paths are NOT understood). Device names are however, and if you are using TOPS10 version 7.01 you will find that using logical names is a good fix - see the PATH system command). It is not known whether this facility will work under TOPS-20, as it relies on the TOPS-10 system call RUN.

5. Setting up TMPCOR files

The evaluable predicate

tmpcor(IO, TmpFile, TmpData)

will read/write tmpcor files. IO can be one of {see, tell} but currently only tell (writing tmpcor files) is implemented. TmpFile is an atom with a 3 character name and TmpData a list of ASCII character codes (written easily as a double quoted string "<characters>"). If IO = tell then the ASCII characters are written to the appropriate tmpcor file. If the tmpcor file cannot be set up then a disk file JJJnnn.TMP is used in the usual way (JJJ is your job number, and nnn = TmpFile). This evaluable predicate is intended for use with run/2 since many programs can be given startup instructions via tmpcor files when started at their CCL entry point - ie offset 1 (eg tmpcor('edt', 'Sfoobaz,fl$'), run(teco,1)). See the TOPS10 manuals for more information on tmpcor files and CCL entry. It is not known whether this facility will work under TOPS-20, as it relies on the TOPS-10 system call TMPCOR.

6. Initialisation

The interpreters initialisation sequence has been extended so that it is now possible to restore a previous save state. The sequence is now:

If prolos.bin exists then restore('prolos.bin')
otherwise If prolos.ini exists then consult('prolos.ini')
otherwise start normally.

The evaluable predicate:

reinitialise

can be used to force these startup actions at any time.
This is a useful step after returning from a core_image. E.g;

?- core_image, display('My Program'), ttyp1, reinitialise.

The prolos.bin facility is likely to be useful in conjunction with the run/2 evaluable predicate (see above), especially if you can set the other
Program to run Prolog again when it finishes. For such uses the save/2 evaluable predicate should be used to save the state (into Prolos.bin) since you will need to distinguish returning from the initial save, and the restore after Prolos is re-run.

7. Banners and copyright messages
-------------------------------------

A new convention for such messages has been set up. Prolog will display its own banner when initially run but not at any time after this (unless asked with ‘version’). This allows your programs to display their own banners after restore, core_image and reinitialise.

The evaluable predicate

    version

will display the banners (which probably includes copyright messages) for all the component parts of the current system.

The evaluable predicate

    version(Mess)

adds a new message to Prolog’s list of messages for component parts. Mess should be an atom. The standard Prolog starts with only one component part message (its own), but users can use version(Mess) to add their own as they build on top of previous systems. Note that this process is always incremental (messages cannot be removed).

8. Extra options for ^C and debussing
----------------------------------------

A few options have been added to the ^C handler and the debussing packages as follows:

    ^C handler
    ^a  - Will try and abort the interpreter as soon as possible

    Debussing
    ^x  - The ‘x’ option should now work better (but not yet completely optimally)
    ^;  - redo; At an EXIT port, force a move to the REDO port
    ^C  - Consult user

9. Break levels
-----------------

The interpreter now indicates your current break level (ie depth of nested breaks) by printing the break level before the final yes/no response to questions, E.g., at break level 2 this would look like:

    ! ?- true.
    [2] yes
    ! ?-

At level 0 (top level) this indicator is omitted (and output looks the same as in previous versions).

10. Image Terminal Output
It is now possible to output image mode (8 bits) characters to the controlling terminal. Any call

\texttt{ttput(C)}

with \(C > 127\) or \(C < 0\) will output the 8 low order bits of \(C\) to the terminal, bypassing the log file, and without buffering. Thus the following call will send all characters \(C\), without altering or losing them:

\texttt{ttput(8'300000+C)}

Although, in theory, this facility makes the new system incompatible with previous ones, it is not expected that many existing programs will hit the incompatibility. The purpose of image mode terminal output is to make it possible to write certain graphics applications in Prolog. This way of implementing it is of course an unprincipled kludge, but it was felt that there wasn't enough demand for the facility to justify monopolising useful predicate names for this purpose.

--- Print and Portray
---

The pretty print procedure ‘Print’ has been redefined so that it will call \texttt{portray} at every level of the term being printed. So now \texttt{print(X)} will:

- write out \(X\) if \(X\) is a variable (so portrays never have to catch variables),
- otherwise call \texttt{portray(X)}
  - if this succeeds then assume \(X\) has been printed
  - else recurse and print the rest of the term
    (unless \(X\) is atomic, in which case just write \(X\)).

So now \texttt{print} can be left to print most of the term, handling brackets, operators and so forth nicely — and you can just catch the bits you want done specially with \texttt{portray}.

Note that on lists ([...]) \texttt{print} will first give the whole list to \texttt{portray}, but after this it will only give each of the (top level) elements to \texttt{portray}. I.e., \texttt{portray} will not get all the tails thrown at it (this could be described as a ‘MAKEAR’ effect).
@comment{ PRESS.BIB - Bibliography file for the Press Project }

@strings(DAI="Dept. of Artificial Intelligence, Edinburgh")
@strings(IJCAI="International Joint Conference on Artificial Intelligence")
@strings(IRIA="Institut de Recherche d'Informatique et d'Automatique")
@strings(AISB="Society for the Study of Artificial Intelligence and Simulation of Behaviour")

@comment{Marker for AAAaaa}

@inproceedings(aubin,
key = "Aubin",
author = "Aubin, R.",
title = "Some generalization heuristics in proofs by induction",
booktitle = "Actes du Colloque Construction! Amelioration et verification de Programmes",
organization = "",
editor = "Huet, G. and Kahn, G.",
year = 1975)

@comment{Marker for BBBbbb}

@book(barnard,
key = "Barnard and Child",
author = "Barnard and Child",
title = "Higher Algebra",
publisher = "MacMillan",
year = 1936)

@book(bartlett,
key = "Bartlett",
author = "Bartlett",
title = "Rememberings",
publisher = "Cambridge Univ. Press",
year = 1967)

@inproceedings(campbell,
key = "Belovari & Campbell",
author = "Belovari, G. and Campbell, J.A.",
title = "Generating contours of integration: an application of PROLOG in symbolic computing",
booktitle = "5th Conference on Automated Deduction",
organization = "Springer Verlag",
editor = "Bibel, W. and Kowalski, R.",
year = 1980,
note = "Lecture Notes in Computer Science No. 87")

@inproceedings(kanoui,
key = "Bersman & Kanoui",
author = "Bersman, M. and Kanoui, H.",
title = "Application of mechanical theorem provers to symbolic calculus",
booktitle = "Third International Symposium on Advanced Computing Methods in Theoretical Physics",
organization = "C.N.R.S.",
year = 1973)

@inbook(berliner,
key = "Berliner",
author = "Berliner, H.J.",
year = 1974)
@techreport{Bledsoe-inf,
  key = "Bledsoe",
  author = "Bledsoe, W.W.",
  title = "The Surf-Inf method in Presburger Arithmetic",
  institution = "Math. Dept., U. of Texas",
  year = 1974,
  number = "ATP-18",
  type = "Memo",
  month = "Dec")

@article{bledsoe-survey,
  key = "Bledsoe",
  author = "Bledsoe, W.W.",
  title = "Non-Resolution theorem-proving",
  journal = "Artificial Intelligence",
  volume = 9,
  year = 1977,
  pages = "1-35",
  number = 1,
  month = "August")

@inproceedings{bledsoe,
  key = "Bledsoe and Bruell",
  author = "Bledsoe, W.W. and Bruell, F.",
  title = "A man machine theorem proving system",
  organization = "Stanford",
  booktitle = "Proc of IJCAI'3",
  year = 1973,
  editor = "Nilsson, N.",
  pages = "56-65")

@techreport{Bledsoe-Hines,
  key = "Bledsoe & Hines",
  author = "Bledsoe, W.W. and Hines, L.M.",
  title = "Variable elimination and chaining in a resolution-based
  prover for inequalities",
  institution = "Math. Dept., U. of Texas",
  year = 1980,
  number = "ATP-56a",
  type = "Memo",
  month = "April")

@PhDThesis{bobrow,
  key = "Bobrow",
  author = "Bobrow, D.",
  title = "Natural Language input for a computer problem solving system",
  school = "MIT",
  year = 1964")

@article{sus,
  key = "Bobrow et al",
  author = "Bobrow, D.G., Kaplan, R.M., Norman, D.A., Thompson, H.
  and Winograd, T.",
  title = "Gus, a frame-driven dialog system",}
note = "Longer version available from Edinburgh as DAI Research Paper No. 159",
year = "1981"

@techreport(induct,
  key = "Bundy & Silver",
  author = "Bundy, A. and Silver, B.",
  title = "A critical survey of rule learning programs",
  institution = DAI,
  year = 1981,
  number = 169,
  type = "Research Paper",
  note = "To appear in Proceedings of ECAI, 1982"
)

@comment(note = "Submitted to the Artificial Intelligence Journal")

@techreport(impress,
  key = "Bundy and Sterling",
  author = "Bundy, A. and Sterling L.S.",
  title = "Meta-level Inference in Algebra",
  institution = DAI,
  month = "September",
  year = "1981",
  type = "Research Paper",
  number = 164,
  note = "Presented at the workshop on logic programming for intelligent systems, Los Angeles, 1981"
)

@techreport(press,
  author = "Bundy, A. and Welham, B.",
  key = "Bundy and Welham",
  title = "Using meta-level descriptions for selective application of multiple rewrite rules in algebraic manipulation",
  institution = DAI,
  year = 1979,
  type = "Working Paper",
  number = 55,
  month = "May"
)

@article(meta,
  author = "Bundy, A. and Welham, B.",
  key = "Bundy and Welham",
  title = "Using meta-level inference for selective application of multiple rewrite rules in algebraic manipulation",
  journal = "Artificial Intelligence",
  year = 1981,
  volume = 16,
  number = 2
)

@book(burnside,
  key = "Burnside and Panton",
  author = "Burnside, W.S. and Panton, A.W.",
  title = "The theory of equations",
  publisher = "Longmans, Green & Co.",
  year = 1881
)

@inproceedings(byrd,
  key = "Byrd",
  author = "Byrd, L.",
  title = "Understanding the control flow of PROLOG programs",
  organization = ""
)

booktitle = "Proceedings of the Logic Programming Workshop",
year = 1980,
editor = "Tarnlund, S.",
pages = "127-38")

@InProceedings(byrd-borning,
key = "Byrd and Borning",
author = "Byrd, L. and Borning, A.",
title = "Extending Mecso to Solve Statics Problems",
organization = AISB,
booktitle = "Proceedings of AISB-80",
year = "1980",
editor = "Hardy, S.",
note = "Also available from Edinburgh as DAI Research Paper No. 137")

@techreport(version3,
key="Byrd et al",
author="Byrd L., Pereira F and Warren D",
title="A guide to Version 3 of DEC-10 Prolog",
institution="Dept of Artificial Intelligence, University of Edinburgh",
number="Occasional Paper 19",
year=1980,
month="Jul")

@comment(Marker for CCCccc)

@book(cardan,
key = "Cardan",
author = "Cardano G",
title = "The Great Art or the rules of Algebra",
publisher = "The MIT Press",
year = 1968,
note = "Translated from the Italian (Ars Magna 1545) by Witmer, T.R")

@unpublished(carry,
key = "Carry et al",
author = "Carry L.R., Bernard, J. and Lewis, C.",
title = "Psychology of Equation Solving: An information processing study",
note = "A grant proposal to NSF in which it is proposed to use PRESS as the basis of a Psychologisical model",
year = 1978)

@book(chang-lee,
key = "Chang and Lee",
author = "Chang C-L. and Lee R. C-T.",
title = "Symbolic logic and mechanical theorem provins",
publisher = "Academic Press",
year = 1973)

@inproceedings(charniak,
key = "Charniak",
author = "Charniak, E.",
title = "Computer solution of calculus word problems",
organization = "IJCAI",
booktitle = "Proc of the 1st",
year = "1969",
editor = "Walker, B.E. and Norton, L.M.",
pages = "303-316")

@article(lambda,
title = "Elementary Numerical Analysis",
publisher = "McGraw-Hill Kosakusha",
year = "1972"

@inproceedings(cooper,
    key = "Cooper",
    author = "Cooper, D.C.",
    title = "Theorem proving in arithmetic without multiplication",
    publisher = "Elsevier, New York",
    booktitle = "Mach, Intell. 7",
    year = 1972,
    editor = "Meltzer, B. and Michie, D.",
    pages = "pp 91-99"
)

@unpublished(cotton,
    key = "Cotton",
    year = 1977,
    note = "Letter describing plans to include instruction in PRESS Processes in remedial Mathematics CAI system",
    author = "Cotton, J.W.",
    title = "Personal communication"
)

@techreport(cotton-evans,
    key = "Cotton et al",
    author = "Cotton, J., Byrd, L. and Bundy, A.",
    title = "How can Algebra steps be learned by students with only arithmetic skills",
    institution = "DAI",
    year = 1981,
    number = 84,
    type = "Working Paper"
)

@comment{Marker for DDDddd}

@techreport(dahl,
    key = "Dahl & Sambuc",
    author = "Dahl, V. and Sambuc, R.",
    title = "Un systeme de bases de donnees en Logique du Premier Ordre, en vue de sa consultation en langue naturelle",
    institution = "Universite d'Aix Marseille",
    year = 1976,
    type = "Rapport"
)

@article(darlington,
    key = "Darlington",
    author = "Darlington J.",
    title = "An Experimental Program Transformation and Synthesis System",
    journal = "Artificial Intelligence",
    volume = 16,
    year = 1981,
    pages = "1-46",
    number = 3,
    month = "August"
)

@article(mycin,
    key = "Davis et al",
    author = "Davis, R., Buchanan, B.G. and Shortliffe, E.H.",
    title = "Production rules as a representation for a knowledge-based consultation program",
    journal = "Artificial Intelligence",
)
@inproceedings(teiresias,
key = "Davis et al",
author = "Davis, R. and Buchanan, B.G.",
title = "Meta-level knowledge: overview and applications",
organization = "IJCAI",
booktitle = "Proc of 5th",
year = "1977",
editor = "Reddy, R.",
pages = "920-927")

@techreport(logomrimer,
key = "du Boulay and O'Shea",
author = "du Boulay, B. and O'Shea, T.",
title = "How to work the LOGO machine",
institution = "DAI",
year = 1976,
number = 4,
month = "November",
type = "Occasional Paper")

@inproceedings(metaprimer,
key = "du Boulay and O'Shea",
author = "du Boulay, B. and O'Shea, T.",
title = "Screening the works: A strategy for teaching interactive programming",
organization = "",
year = 1978,
address = "Liverpool",
month = "March",
note = "also available as DAI working paper no. 28")

@article(duboulay,
key = "du Boulay",
author = "du Boulay, B.",
title = "Teaching teachers mathematics through programming",
journal = "International Journal of Mathematics Education in Science and Technology",
year = 1979,
note = "also available as DAI Research Paper no. 113")

@TechReport(dekleer,
key = "De Kleer",
author = "De Kleer, J.",
title = "Qualitative and quantitative knowledge in classical mechanics",
institution = "MIT AI Lab",
year = "1975",
type = "AI-TR-352")

@book(brainstorms,
key = "Dennett",
author = "Dennett, D.C.",
title = "Brainstorms: Philosophical Essays on Mind and Psychology",
publisher = "Harvester Press"
@article{evans,
  key = "Evans",
  author = "Evans, T.G.",
  title = "A heuristic program to solve geometric analogy problems",
  journal = "J.S.C.C.",
  year = 1964,
  month = "April"
}

@PhDThesis{fateman,
  key = "Fateman",
  author = "Fateman, R.J.",
  title = "Essays in Algebraic Simplification",
  school = "MIT",
  month = "April",
  year = "1972",
  note = "also available as MAC TR-95"
}

@InProceedings{feisenbaum,
  key = "Feisenbaum",
  author = "Feisenbaum, E. A.",
  title = "Themes and case studies of Knowledge Engineering",
  publisher = "Edinburgh Univ. Press",
  booktitle = "Expert systems in the micro-electronic age",
  year = 1979,
  editor = "Michie, D.",
  pages = "3-25"
}

@techreport{friend,
  key = "Friend",
  author = "Friend, J.",
  title = "Programs students write",
  institution = "Stanford University",
  year = 1975,
  number = 257
}

@MastersThesis{funt,
  key = "Funt",
  author = "Funt, B. V.",
  title = "A procedural approach to constructions in Euclidean geometry",
  school = "University of British Columbia",
  month = "October",
  year = 1973
}

@InProceedings{futo,
  key = "Futo et al",
  author = "Futo, I., Darvas, F. and Szerebi, P.",
  title = "The application of PROLOG to the development of QA and DBM systems",
  booktitle = "Logic and databases",
  organization = "Plenum Press, New York",
  pages = "pp 347-375",
  editor = "Gallaire, H. and Minker, J.",
  year = 1978"}
@inbook{Gelernter,  
key = "Gelernter",  
author = "Gelernter, H.",  
title = "Realization of a Geometry theorem-proving",  
publisher = "McGraw Hill",  
year = 1963,  
editor = "Feisenbaum and Feldman",  
booktitle = "Computers and Thought",  
pages = "134-52"
}

@inbook{Gelernter2,  
key = "Gelernter et al",  
author = "Gelernter, H. et al",  
title = "Empirical explorations of the Geometry theorem-proving machine",  
publisher = "McGraw Hill",  
year = 1963,  
editor = "Feisenbaum and Feldman",  
booktitle = "Computers and Thought",  
pages = "153-63"
}

@article{Gilmore-method,  
key = "Gilmore",  
author = "Gilmore, P.C.",  
title = "A proof method for quantificational theory",  
journal = "IBM J Res. Dev.",  
year = 1960,  
pages = "28-35",  
volume = 4
}

@article{Gilmore,  
key = "Gilmore",  
author = "Gilmore, P.C.",  
title = "An examination of the Geometry theorem-proving machine",  
journal = "Artificial Intelligence",  
volume = 1,  
year = 1970,  
pages = "171-87"
}

@PhDThesis{Goldbergs,  
key = "Goldbergs",  
author = "Goldbergs, A.",  
title = "Computer assisted instruction: The application of theorem proving to adaptive response analysis",  
school = "Stanford",  
month = "May",  
year = 1973,  
ote = "Also published as IMSSS Stanford Technical Report 203"
}

@TechReport{Goldstein,  
key = "Goldstein",  
author = "Goldstein, I.",  
title = "Elementary Geometry theorem proving",  
institution = "MIT",  
year = 1973,  
type = "AI TechReport",  
number = 280
}

@article{Good,
key = "Good & London",
title = "Computer interval arithmetic: definition and proof of correct implementation",
journal = "JACM",
year = 1970,
volume = 17,
number = 4,
pages = "603-612"

@book(lcf,
key = "Gordon et al",
title = "Edinburgh LCF - A mechanised logic of computation",
publisher = "Springer Verlag",
year = 1979,
series = "Lecture Notes in Computer Science",
volume = 78)

~comment{Marker for HHHhhh}

@book(halliday,
key = "Halliday",
author = "Halliday, D. and Resnick R.",
title = "Physics",
publisher = "John Wiley and Sons",
year = "1966")

@MastersThesis(hammond,
key = "Hammond",
author = "Hammond, P.",
title = "Logic Programming for Expert Systems",
school = "Imperial College, London",
year = "1980")

@article(haviland,
key = "Haviland et al",
author = "Haviland, S.E. and Clark, H.H.",
title = "What's new? Acquiring new information as a process in comprehension",
journal = "Journal of Verbal Learning and Verbal Behaviour",
volume = 13,
pages = "512-521",
year = 1974)

@inproceedings(solux,
key = "Hayes",
author = "Hayes, P.",
title = "Computation and deduction",
organization = "Czech. Academy of Sciences",
booktitle = "Proc. of MFCS Symposium",
year = 1973)

@inbook(hearn,
key = "Hearn",
author = "Hearn, A.C.",
title = "REDUCE: A user-oriented interactive system for Algebraic simplification",
publisher = "Academic Press, New York",
year = "1967"
booktitle = "Interactive systems for experimental Applied Mathematics",
pages = "79-90")

@incollection(herbrand,
key = "Herbrand",
author = "Herbrand, J.",
title = "Researches in the theory of demonstration",
booktitle = "From Frege to Goedel: a source book in Mathematical Logic, 1879-1931",
publisher = "Harvard Univ. Press",
year = 1930,
address = "Cambridge, Mass",
editor = "van Heijenoort, J.",
pages = "525-81")

@techreport(lishres,
key = "Hill",
author = "Hill, R.",
title = "Lush-Resolution and its completeness",
institution = DAI,
year = 1974,
number = 78,
month = "August",
type = "DCL Memo")

@article(howe,
key = "Howe and O'Shea",
author = "Howe, J.A.M. and O'Shea, T.",
title = "Learning Mathematics through LOGO",
journal = "SIGCUE Bulletin",
year = 1978,
number = 12)

@techreport(huet74,
key = "Huet",
author = "Huet, G.P.",
title = "A unification algorithm for typed lambda-calculus",
institution = IRIA,
year = 1974,
type = note de travail,
number = "A 055",
month = March)

@TechReport(huet,
key = "Huet",
author = "Huet, G",
title = "Confluent reductions: Abstract properties and applications to term rewriting systems",
institution = "Laboratoire de Recherche en Informatique et Automatique, IRIA, France",
month = "August",
type = "Rapport de Recherche",
number = 250,
year = "1977")

@book(humphrey,
key = "Humphrey",
author = "Humphrey, D.",
title = "Intermediate Mechanics, Dynamics",
publisher = "Longman, Green & Co., London")
@book{hofstadter,
  key = "Hofstadter",
  author = "Hofstadter, D.R.",
  title = "Godel, Escher, Bach: An Eternal Golden Braid",
  publisher = "Harvester Press",
  price = "$10.50",
  year = 1979)

@comment{Marker for IIIiii}

@comment{Marker for JJJjjj}

@comment{Marker for KKKkkk}

@book{kleene,
  key = "Kleene",
  author = "Kleene, S.C.",
  title = "Mathematical Logic",
  publisher = "John Wiley & Sons, Inc.",
  year = 1967)

@InCollection{knuth-bendix,
  key = "Knuth and Bendix",
  author = "Knuth, D.E., and Bendix, P.B.",
  title = "Simple word problems in universal algebra",
  booktitle = "Computational problems in abstract algebra",
  publisher = "Pergamon Press",
  year = 1970,
  editor = Leech, J.,
  pages = "pp 263-297")

@TechReport{kowalski,
  key = "Kowalski",
  author = "Kowalski, R.",
  title = "Logic for Problem Solving",
  institution = DAI,
  type = "DCL TechReport",
  number = 75,
  year = 1974)

@book{kowalskibook,
  title = "Logic for Problem Solving",
  author = "Robert Kowalski",
  key = "Kowalski",
  publisher = "North Holland",
  year = 1979)

@TechReport{kowalski-hayes,
  key = "Kowalski and Hayes",
  author = "Kowalski, R., and Hayes, P.",
  title = "Lecture notes on theorem proving",
  institution = DAI,
  year = 1974)

@article{slres,
  key = "Kowalski and Kuehner",
  title = "Linear Resolution with selection function",
  year = 1974)
journal = "Artificial Intelligence",
year = 1971,
volume = 2,
pages = "227-60")

@comment{Marker for LLLL11}

@book{lakatos,
key = "Lakatos",
author = "Lakatos, I.",
title = "Proofs and refutations: The logic of Mathematical discovery",
publisher = "Cambridge University Press",
year = 1976
}

@techreport{lansley,
key = "Lansley",
author = "Lansley, P.",
title = "Language acquisition through error recovery",
institution = "Carnegie-Mellon University",
year = 1981,
number = 432,
month = "June",
type = "CIP Working Paper"
}

@TechReport{larkin,
key = "Larkin",
author = "Larkin, J.",
title = "Problem solving in Physics",
institution = "Group in Science and Mathematics Education, Berkeley, California",
year = "1977"
}

@TechReport{larkinMcD,
key = "Larkin",
author = "Larkin, J.H. and McDermott, J.",
title = "Re-representing Textbook Physics Problems",
institution = "Carnegie Mellon University",
year = "1978"
}

@techReport{larkin2,
key = "Larkin",
author = "Larkin, J.H.",
title = "Models of Strategy for Solving Physics Problems",
institution = "Carnegie Mellon University",
year = "1979"
}

@PhDThesis{lenat,
key = "Lenat",
author = "Lenat D.B.",
title = "AM: An Artificial Intelligence approach to discovery in Mathematics as Heuristic Search",
school = "Stanford",
month = "July",
year = 1976,
ote = "Available from Stanford as TechReport AIM 286"
}

@book{loveland,
key = "Loveland",
author = "Loveland, D.W.",
title = "Automated theorem proving: A logical basis",
year = "1978"
}
author = "Marple, D.",
title = "Arsument and technique in the solution of problems in Mechanics and Electricity",
institution = "Dept. of Engineerings, Cambridge, England",
year = "1974",
type = "CUED/C-Educ/TRI"

@inproceedings(martin,
key = "Martin and Fateman",
author = "Martin, W.A. and Fateman, R.J.",
title = "The MACSYMA system",
organization = "Los Angeles",
booktitle = "2nd Symposium on Symbolic Manipulation",
year = "1971",
editor = "Petrick, S.R.",
pages = "59-75"
)

@techreport(mellish,
author = "Mellish, C.S.",
institution = "Dept of Artificial Intelligence, University of Edinburgh",
key = "Mellish",
title = "Preliminary Syntactic Analysis and Interpretation of Mechanics Problems Stated in English",
year = "1978",
number = "48",
type = "Working Paper"
)

@PhDThesis(mellish2,
key = "Mellish",
author = "Mellish, C.S.",
title = "Constructions with uncertainty: Noun phrase interpretation and early semantic analysis",
school = "Dept of Artificial Intelligence, University of Edinburgh",
year = "1980"
)

@inproceedings(meltzer,
key = "Meltzer",
author = "Meltzer, B.",
title = "The use of Symbolic Logic in proving Mathematical theorems by means of a digital computer",
organization = "Springer Verlag",
booktitle = "Foundations of Mathematics! Symposium Papers commemorating the sixtieth birthday of Kurt Godel",
year = "1969",
editor = "Bulloff, J.J., Holyoke, T.C. and Hahn, S.W.",
pages = "39-45",
note = "Also available from Edinburgh as DCL Memo No. 21"
)

@book(mendelson,
key = "Mendelson",
author = "Mendelson, E.",
publisher = "van Nostrand Reinhold Co.",
title = "Introduction to Mathematical Logic",
year = "1964"
)

@PhDThesis(moorethesis,
key = "Moore",
author = "Moore, J.",

title = "Computational Logic: Structure sharing and proof of
Program properties, Part II",
institution = "Univ. of Edinburh",
year = 1974,
note = "Available from Edinburh as DCL memo no. 68 and
from Xerox PARC, Palo Alto as CSL 75-2."

@techreport(merlin,
key = "Moore and Newell",
author = "Moore, J. and Newell, A.",
title = "How can Merlin understand",
institution = "Dept. of Computer Science, Carnesie Mellon University",
year = 1973)

@techreport(milne,
key = "Milne",
author = "Milne, R.",
title = "A Framework for Deterministic Parsing Using Syntax and Semantics",
institution = "Department of Artificial Intelligence, University of
Edinburh",
year = 1980,
number = 64,
type = "Working Paper"
)

@inproceedings(milne2,
key = "Milne",
author = "Milne, R.",
year = 1980,
title = "Using Determinism to Predict Garden Paths",
organization = "AISB",
booktitle = "AISB 80 Conference Proceedings"
)

@inproceedings(milne3,
key = "Milne",
author = "Milne, R.",
year = 1980,
title = "Parsing Against Lexical Ambiguity",
booktitle = "Proceedings for COLING 80",
organization = ""
)

@PhDThesis(Mitchellthesis,
key = "Mitchell",
author = "Mitchell, T.M.",
title = "Version Spaces: An approach to concept learning",
school = "Stanford University",
year = 1978)

@inproceedinss(mitchell,
key = "Mitchell et al",
author = "Mitchell, T.M., Utsoff, P. E., Nudel, B. and Banerji, R.",
title = "Learning problem-solving heuristics through practice",
booktitle = "IJCAI-81",
organization = IJCAI,
pages = "127-134",
year = 1981)

@PhDThesis(sin,
key = "Moses",
author = "Moses, J",
title = "Symbolic integration"
key="Quinlan",
author="Quinlan, J.R.",
title="Discovering Rules by Induction from large collections of examples",
publisher="EUP",
booktitle="Expert Systems in the Micro-Electronic Age",
pages = "168-201",
year=1979,
editor="Michie D")

@comment{Marker for RRRrrr}

@article(raulefs,
  key = "Raulefs et al",
  author = "Raulefs, F., Siekmann, J., Szabo, P. and Unverricht, E.",
  title = "A short survey on the state of the art in matching and unification problems",
  journal = "AISB Quarterly",
  volume = "issue 32",
  year = 1978,
  pages = "pp.17-21",
  month = "December")

@InProceedings(reiter,
  key = "Reiter",
  author = "Reiter, R.",
  title = "A semantically guided deductive system for automatic theorem-proving",
  organization = "IJCAI",
  booktitle = "Proc. of the 3rd",
  year = 1973,
  editor = "Nilsson, N",
  pages = 41-6)

@techreport(default,
  key = "Reiter",
  author = "Reiter, R.",
  title = "A logic for default reasoning",
  institution = "Univ. of British Columbia",
  year = 1979,
  number = "79-8",
  month = "July",
  type = "Technical Report")

@techreport(paraconj,
  key = "Richter",
  author = "Richter, M.",
  title = "A note on paramodulation and the functional reflexive axioms",
  institution = "Univ. of Texas at Austin",
  year = 1974)

@techreport(ritchie,
  key = "Ritchie",
  author = "Ritchie, G.",
  title = "AM: A case study in AI methodology",
  institution = "Univ. of Kent",
  year = 1981)

@article(ripley,
  key = "Ripley and Druseikis",
  author = "Ripley, G.D. and Druseikis, F.C.",
A statistical analysis of syntax errors,
journal = "Computer Languages",
year = 1978,
number = 3,
pages = "227-240"

@article(resolution,
  key = "Robinson",
  author = "Robinson, J.A.",
  title = "A machine oriented logic based on the Resolution Principle",
  journal = "J Assoc. Comput. Mach.",
  year = 1965,
pages = "23-41",
  volume = 12)

@book(robinson,
  key = "Robinson",
  author = "Robinson, J.A.",
  title = "Logic: Form and function. The mechanization of deductive reasoning",
  publisher = "Edinburgh University Press",
  year = 1979)

@incollection(paramodule,
  key = "Robinson and Wos",
  author = "Robinson, G. and Wos, L.",
  title = "Paramodulation and Theorem-provins in first-order theories with equality",
  publisher = "Edinburgh University Press",
  booktitle = "Machine Intelligence 4",
  year = 1969,
  pages = "pp103-33")

@techreport(roussel,
  key = "Roussel",
  author = "Roussel, P.",
  title = "PROLOG : Manuel de reference et d’utilisation",
  institution = "Universite d’Aix Marseille",
  year = 1975,
  type = "Rapport")

@comment{Marker for SSSssss}

@article(scandura,
  key = "Scandura",
  journal = "Artificial Intelligence",
  title = "Higher order rule characterization of heuristics for compass and straight edge constructions in Geometry",
  year = 1974,
  volume = 5,
  pages = "pp 149-184")

@unpublished(schoenfield,
  key = "Schoenfield",
  author = "Schoenfield, A.H.",
  year = 1979,
  note = "Letter expressing interest in PRESS as a teachable model of"
expert problem solvin3',
title = "Personal communication")

@article(schubert,
key = "Schubert",
author = "Schubert, L.K.",
title = "Extending the expressive power of Semantic Networks",
journal = "Artificial Intelligence",
year = 1976,
volume = 7,
pages = "pp 89-124")

@inproceedings(shapiro,
key = "Shapiro",
author = "Shapiro, E.",
title = "An algorithm that infers theories from facts",
booktitle = "IJCAI-81",
organization = "IJCAI",
pages = "446-451",
year = 1981)


@inproceedings(key = "Tyusu", title = "A Computational Problem-solver", year = 1979, publisher = """, editors = "Michie, D. et al.", booktitle = "Machine Intelligence 9")

@article(key = "Waerden", title = "Wie der beweis der vermutung von Baudet gefunden wurde", year = "", journal = "Abh. Math. Sem. Univ. Hamburg", volume = 28, note = "I possess a photocopy of an English translation entitled 'How the proof of Baudet's conjecture was found', This was given to me by Gordon Plotkin who, unfortunately, has forgotten the source.")

@TechReport(key = "Waltz", title = "Generating semantic descriptions from drawings of scenes..."
year = 1961

@TechReport{fol,  
  author = "Weyhrauch, R.W.",  
  title = "Prolegomena to a theory of mechanized formal reasoning",  
  key = "Weyhrauch",  
  institution = "Stanford University",  
  type = "RWW Informal Note 8.",  
  year = "1979" }

@book{wino,  
  key = "Winstorad",  
  author = "Winstorad, T",  
  title = "Understanding Natural Language",  
  publisher = "Edinburgh University Press",  
  year = "1972"
}

@inbook{winston,  
  key = "Winston",  
  author = "Winston, P.",  
  title = "Learning structural descriptions from examples",  
  booktitle = "The Psychology of Computer Vision",  
  editor = "Winston, P.H.",  
  publisher = "McGraw Hill",  
  year = 1975
}

@inproceedings{robinson-wos,  
  key = "Wos et al",  
  author = "Wos, L., Robinson, G. and Carson, D.F.",  
  title = "The automatic generation of proofs in the language of Mathematics",  
  organization = "IFIP",  
  booktitle = "IFIP Congress 65",  
  year = 1965
}

@inproceedings{wos-cade6,  
  key = "Wos",  
  author = "Wos, L.",  
  title = "Solving open questions with an automated theorem-proving program",  
  organization = "",  
  booktitle = "Proceedings of CADE6",  
  year = 1982,  
  editor = "Loveland, D."}

@comment{Marker for XXXxxx}

@comment{Marker for YYYYyy}

@article{yohe,  
  key = "Yohe",  
  author = "Yohe, J.M.",  
  title = "Software for interval arithmetic: A reasonably portable package",  
  journal = "ACM Trans. Math. Software",  
  year = 1979,  
  number = 1,  
  month = March,  
  pages = "pp50-63",  
  volume = 5"}
@inproceedings(plotkin-young,
  key = "Youn et al",
  author = "Young, R.M., Plotkin, G.D. and Linz, R.F.",
  title = "Analysis of an extended concept-learning task",
  booktitle = "IJCAI-77",
  organization = IJCAI,
  pages = "p285",
  editor = "Reddy, R.",
  year = 1977)

@article(youngs,
  key = "Youngs",
  author = "Youngs, E.A.",
  title = "Human errors in programming",
  journal = "Int J Man-Machine Studies",
  year = 1974,
  number = 6,
  pages = "361-376")

^comment{Marker for ZZZzzz}
PROLOG BIBLIOGRAPHY

This short bibliography concentrates on work done in Edinburgh. It does not try to cover the wealth of material from the rest of Europe and the USA.

Prolog - systems/implementations

[1] Byrd L, Pereira F and Warren D.
A guide to Version 3 of DEC-10 Prolog.
Occasional Paper 19, Dept. of Artificial Intelligence,

Understanding the control flow of PROLOG programs.
In Tarnlund, S., editor, Proceedings of the Logic Programming
Also available as DAI Research Paper 150.

[3] Clocksin W F and Mellish C S.
The UNIX Prolog System.
Software Report 5, Dept. of Artificial Intelligence, Edinburgh,
September, 1980.

[Information about EMAS Prolog is available from its author].
Dept of Computer Science, University of Edinburgh.

An alternative to structure sharing in the implementation of a
Prolog interpreter.
In Tarnlund, S., editor, Proceedings of the Logic Programming
Workshop, , 1980.
Also available as DAI Research Paper 151.

User’s Guide to DECsystem-10 PROLOG.
Occasional Paper 15, Dept. of Artificial Intelligence,

[7] Roberts G M.
An implementation of Prolog.
Master’s thesis, Dept of Computer Science, Univ of Waterloo,

Higher-order extensions to Prolog - are they needed?.
In Tenth International Machine Intelligence Workshop, Cleveland,
Ohio, , Apr, 1981.
Also available as DAI Research Paper 154.
Warren D H D, Pereira L M and Pereira F.
Prolog - the language and its implementation compared with Lisp.
In ACM Symposium on AI and Programming Languages,
SIGPLAN/SIGART Newsletter, August, 1977.

Implementing Prolog - compiling predicate logic programs.
Dept. of Artificial Intelligence, Edinburgh, 1977,
Research Reports 39 & 40.

Prolog on the DECSYSTEM-10.

Warren D.
An Improved Prolog Implementation which optimises tail recursion.
Also available as DAI Research Paper 141.

Clocksin, W.F, and Mellish, C.S.
Programming in Prolog,
, in preparation.

Coelho H, Cotta J C and Pereira L M.
How to solve it with Prolog.
Laboratorio Nacional de Engenharia Civil, Lisbon, 1980.

Kowalski R A.
Algorithm = logic + control.

Robert Kowalski.
Logic for Problem Solving.

Tarnlund S-A (ed.),
Proceedings of the 1980 Logic Programming Workshop, Debrecen,
Hungary,
[Dept of Computer Science, University of Stockholm].

Warren D H D.
Logic programming and compiler writing.
Some applications at Edinburgh

[1] Bundy, A. and Welham, B.
Using meta-level inference for selective application of multiple
rewrite rules in algebraic manipulation.
Artificial Intelligence, in press 1981.
Also available as DAI Research Paper 121.

Solving mechanics problems using meta-level inference.
In Proceedings of the 6th, pages 1017-1027, International Joint
Conference on Artificial Intelligence, 1979.
Also available as DAI research paper 112.

Definite clause grammars for language analysis - a survey of the
formalism and a comparison with augmented transition
networks.
Also available as DAI Research Paper 116.

An efficient easily adaptable system for interpreting natural
language queries.
Research Paper 156, Dept. of Artificial Intelligence, Edinburgh,
1981.
[To be submitted to IJCAI-81].

Efficient processing of interactive relational database queries
expressed in logic.
Research Paper 156, Dept. of Artificial Intelligence, Edinburgh,
1981.
[To be submitted to the 1981 Conf on Very Large Databases].

Geometry Problem Solving.
Intelligence, Edinburgh, 1976.
@Comment[ PROLOG BIBLIOGRAPHY ]

@strings(
eu = "University of Edinburgh",
eup = "Edinburgh University Press",
here = "Dept. of Artificial Intelligence, Edinburgh",
DAI = "Dept. of Artificial Intelligence, Edinburgh",
marsai = "Groupe d'Intelligence Artificielle",
luminy = "U. E. R. de Luminy, Universite d'Aix-Marseille II",
IJCAI = "International Joint Conference on Artificial Intelligence")

@Comment[ Most directly relevant papers ]

@techreport(prolos10,
    key = "Pereira, Pereira and Warren",
    author = "Pereira, L., Pereira, F. and Warren, D.",
    title = "User's Guide to DECsystem-10 PROLOG",
    institution = DAI,
    year = 1979,
    type = "Occasional Paper",
    number = 15)

@techreport(version3,
    key = "Byrd et al",
    author = "Byrd L, Pereira F and Warren D",
    title = "A guide to Version 3 of DEC-10 Prolos",
    institution = DAI,
    type = "Occasional Paper",
    number = 19,
    year = 1980,
    month = "July")

@proceedings(rochester,
    key = "Warren et al",
    title = "Prolog - the language and its implementation compared with Lisp",
    author = "Warren D H M, Pereira L M and Pereira F",
    booktitle = "ACM Symposium on AI and Programming Languages",
    publisher = "SIGPLAN/SIGART Newsletter",
    year = 1977,
    month = "Aug")

@inproceedings(prolosonthe10,
    key = "Warren",
    author = "Warren D H M",
    title = "Prolog on the DECsystem-10",
    publisher = "Eup",
    booktitle = "Expert Systems in the Micro-Electronic Age",
    year = 1979,
    editor = "Michie D")

@book(ip,
    key = "Warren",
    author = "Warren D H M",
    title = "Implementing Prolog - compiling predicate logic programs",
    publisher = "here",
    note = "Research Reports 39 & 40",
    year = 1977,
    month = "May")
@article(warren,  
key="Warren",  
author="Warren D H D",  
title="Logic Programming and compiler writing",  
journal="Software Practice & Experience",  
volume=10,  
pages="97-125",  
year=1980)

@inproceedings(tailrec,  
key = "Warren",  
author = "Warren, D",  
title = "An Improved Prolog Implementation which optimises tail recursion",  
organization = ",  
booktitle = "Proceedings of the Logic Programming Workshop",  
year = 1980,  
editor = "Tarnlund, S.",  
note = "Also available as DAI Research Paper 141")

@inproceedings(alternative,  
key = "Mellish",  
title = "An alternative to structure sharing in the implementation of  
a Prolog interpreter",  
author = "Chris Mellish",  
organization = ",  
booktitle = "Proceedings of the Logic Programming Workshop",  
year = 1980,  
editor = "Tarnlund, S.",  
note = "Also available as DAI Research Paper 151")

@inproceedings(flow,  
key = "Byrd",  
author = "Byrd, L.",  
title = "Understanding the control flow of PROLOG programs",  
organization = ",  
booktitle = "Proceedings of the Logic Programming Workshop",  
year = 1980,  
editor = "Tarnlund, S.",  
pages = "127-38",  
note = "Also available as DAI Research Paper 150")

@inproceedings(MI10,  
key="Warren D H D",  
author="Warren D H D",  
title="Higher-order extensions to Prologs - are they needed?",  
booktitle="Tenth International Machine Intelligence Workshop,  
Cleveland, Ohio",  
publisher="",  
year=1981,  
month="Apr",  
note = "Also available as DAI Research Paper 154")

@misc(debrencen,  
key="Tarnlund",  
author="Tarnlund S-A (ed.)",  
title="Proceedings of the 1980 Logic Programming Workshop",  
Debrecen, Hungary",  
howpublished="[Dept of Computer Science, University of Stockholm]",  
year=1980,
@article(alc,
  key="Kowalski",
  author="Kowalski R A",
  title="Algorithm = logic + control",
  journal="CACM",
  volume=22,
  number=7,
  pages="424-436",
  year=1979,
  month="Jul")

@book(KowalskiBook,
  title = 'Logic for Problem Solving',
  author = 'Robert Kowalski',
  key = "Kowalski",
  publisher = "North Holland",
  year = 1979)

@book(solveit,
  key="Coelho et al",
  author="Coelho H, Cotta J C and Pereira L M",
  title = 'How to solve it with Prolog',
  publisher="Laboratorio Nacional de Engenharia Civil, Lisbon",
  year=1980)

@book(primer,
  key = "Clocksin and Mellish",
  author = "Clocksin, W.F, and Mellish, C.S.",
  title = "Programming in Prolog",
  year = "in preparation",
  publisher = "")

@Comment[ Prolog & NL in Edinbursh ]

@article(dcss,
  key="Pereira & Warren",
  author="Pereira F C N and Warren D H D",
  title="Definite clause grammars for language analysis - a survey of the formalism and a comparison with augmented transition networks",
  journal="Artificial Intelligence",
  volume=13,
  pages="231-278",
  year=1980,
  note="Also available as DAI Research Paper 116")

@inproceedings(xss,
  key="Pereira",
  title="Extraposition Grammars",
  author="Pereira F",
  Booktitle="Proceedings of the Logic Programming Workshop",
  editor="Stan-Ake Tarnlund",
  year=1980,
  month="Jul",
  note="Held in Debrecen, Hungary")
author = "Damas L",
title = "Information about EMAS Prolog is available from its author",
howpublished = "Dept of Computer Science, University of Edinburgh")

@book(fortran,
key = "Battani and Meloni",
author = "Battani G and Meloni H",
title = "Interpr\^{e}teur du langage de Programmation Prolog",
publisher = "Marsais",
address = "luminy",
year = 1973)

@book(bruynooshe,
key = "bruynooshe",
author = "bruynooshe M",
title = "An interpreter for predicate logic programs : Part 1",
publisher = "Applied Maths & Programming Division",
address = "Katholieke Univ Leuven, Belgium",
year = 1976,
month = "oct",
ote = "Report CW 10")

@Comment[ More remote papers ]
@techreport(mastermind,
key = "Emden",
author = "van Emden M H",
title = "Relational Programming illustrated by a Program for the same of Mastermind",
institution = "Dept. of Computer Science, University of Waterloo",
number = "Report CS-78-48",
year = 1978)

@techreport(permont,
key = "Pereira and Monteiro",
author = "Pereira L M and Monteiro L F",
title = "The semantics of parallelism and co-routining in logic programming",
institution = "Div. Informatica, L.N.E.C., Lisbon",
number = "Proc. 03/13/5570",
year = 1978,
ote = "Presented at Colloquium on Mathematical Logic in Programming, Salsotarjan, Hungary, September 1978")

@techreport(lush,
key = "Hill",
author = "Hill R",
title = "LUSH resolution and its completeness",
institution = "DAI",
number = "DCL Memo 78",
year = 1974)

@techreport(plasspI,
author = "Kowalski R A",
title = "Predicate Logic as Programming Language",
key = "Kowalski",
institution = "DAI",
number = "DCL Memo 70",
year = 1973,
ote = "Appears in Procs. IFIP 1977")
@techreport(alsol68moss,
  author = "Moss C D S",
  title = "A New Grammar for Alsol 68",
  key = "Moss",
  institution = "Dept. of Computing and Control, Imperial College, London",
  number = "Report 79/6",
  year = 1979)

@techreport(warrplan,
  author = "Warren D H D",
  title = "A System for Generating Plans",
  key = "Warren",
  institution = DAI,
  number = "DCL Memo 76",
  year = 1974)

@techreport(earleyded,
  author = "Warren D H D",
  title = "Implementation of an Efficient Predicate Logic Interpreter Based on Earley Deduction",
  key = "Warren",
  institution = DAI,
  number = "Research Proposal to the Science Research Council",
  year = 1975)

@techreport(emden75,
  author = "van Emden M H",
  title = "Programming with Resolution Logic",
  key = "Emden",
  institution = "Dept. of Computer Science, University of Waterloo",
  number = "Report CS-75-30",
  year = 1975)

@techreport(emden77,
  author = "van Emden M H",
  title = "Computation and Deductive Information Retrieval",
  key = "Emden",
  institution = "Dept. of Computer Science, University of Waterloo",
  number = "Report CS-77-16",
  year = 1977)

@techreport(andreka,
  author = "Andreka H and Nemeti I",
  title = "The Generalised Completeness of Horn Predicate Logic as a Programming Language",
  key = "Andreka and Nemeti",
  institution = DAI,
  number = "Research Report 21",
  year = "1976")

@book(dwissins,
  key="Dwissins",
  author="Dwissins D L",
  title="A knowledge-based automated message understanding methodology for an advanced indications system",
  publisher="Operating Systems Inc, 21031 Ventura Boulevard, Woodland Hills, California 91364",
  year=1979,
  month=feb,
address=luminy,
year=1977,
month=oct,
note=[To appear in CACM]

@misc(darvas,
key="Darvas et al.",
author="Darvas F, Futo I and Szeredi P",
title="Logic based program system for predicting drug interactions",
howpublished="Int. J. of Biomedical Computing, 1977")

@inproceedings(markusz,
key="Markusz",
author="Markusz Z",
title="Designing variants of flats",
organization="IFIP Conference",
year=1977)

@article(unification,
key="Robinson J A",
author="Robinson J A",
title="A machine-oriented logic based on the resolution principle",
journal="JACM",
volume=12,
number=1,
pages="227-234",
year=1965,
month=dec)

@mastersthesis(scheme,
key="Steele",
author="Steele G L",
title="RABBIT: A Compiler for SCHEME",
school="MIT",
year=1978,
month=may,
address=luminy)

@book(prolos,
key="Roussel",
author="Roussel P",
title="Prolos : Manuel de Reference et d'Utilisation",
publisher=marsai,
year=1975,
address=luminy)

@book(mgs,
key="Colmerauer",
author="Colmerauer, A.",
title="Metamorphosis Grammars",
publisher="Springer-Verlag",
year=1978,
editor="L. Bolc",
booktitle="Natural Language Communication with Computers",
note="First appeared as an internal report, 'Les Grammaires de Metamorphose', in November 1975")

@book(dahl,
key="Dahl",
author="Dahl, V.",
title="Un systeme deductif d'interrosoation de banques de donnees en Espagnol",
publisher=marsai,
year=1977)
@techreport(mecho,
  title = "Mecho: A Program to Solve Mechanics Problems",
  key = "Bundy et al",
  institution = DAI,
  number = "Working Paper No. 50",
  year = "1979"
)

@inproceedings(metamecho,
  author = "Bundy, A., Byrd, L., Luser, G., Mellish, C., Palmer, M.",
  title = "Solving Mechanics Problems Using Meta-level Inference",
  key = "Bundy et al",
  booktitle = "Proceedings of the 6th",
  organization = IJCAI,
  year = 1979,
  note = "Also available as DAI research paper 112"
)

@techreport(grant,
  key = "Welham and Bundy",
  author = "Welham, R. and Bundy, A.",
  title = "Equation Solving: A Progress Report",
  institution = DAI,
  year = "1978",
  note = "Also available as DAI Research Paper 121"
)

@article(fress,
  author = "Bundy, A. and Welham, R.",
  title = "Using Meta-level Inference for Selective Application of Multiple Rewrite Rules in Algebraic Manipulation",
  organization = "Artificial Intelligence",
  year = "in press 1981",
  note = "Also available as DAI Research Paper 121"
)

@techreport(eder,
  title = "A Prolog-like Interpreter for Non-Horn Clauses",
  author = "Eder, Gottfried",
  year = "1976",
  key = "Eder",
  number = "Research Report no. 26",
  institution = DAI
)

@techreport(seom1,
  title = "Geometry Problem Solving",
  author = "Welham, Bob",
  year = "1976",
  key = "Welham",
  number = "Research Report no. 14",
  institution = DAI
)

@techreport(seom2,
  title = "GEOM: A Prolog Geometry Theorem Prover",
  author = "Coelho, H., Pereira, L.M.",
  year = "1976",
  key = "Coelho et al",
  institution = "National Laboratory of Civil Engineering, Lisbon, Portugal"
)
@techreport{Tarnlund,
title = "Logic Information Processing",
author = "Tarnlund, Sten-Ake.",
year = "1975",
key = "Tarnlund",
institution = "Dept. Computer Science, University of Stockholm, Sweden"
)

@inproceedings(Gallaire,
title = "Issues in controlling a deduction process in a declarative mode.",
author = "Gallaire, H., Lasserre, C.",
year = "1979",
key = "Gallaire et al",
organization = IJCAI"
)

@techreport(bruynooshe78,
key = "Bruynooshe",
author = "Bruynooshe M",
title = "Intelligent Backtracking for an Interpreter of Horn Clause Logic Programs",
institution = "Afdeling Toegepaste Wiskunde en Programmatie, Katholieke Universiteit Leuven, Belgium",
number = "Report CW 16",
year = "1978",
note = "Presented at Colloquium on Mathematical Logic in Programming, Salgotarjan, Hungary, September 1978"
)

@techreport(sus,
author = "Mellish, C.",
title = "An approach to the GUS travel agent problem using PROLOG",
year = "1977",
key = "Mellish",
institution = DAI,
number = "Working Paper No. 19",
month = "February"
)
Special letter format for use with HPS large paper. This should be used with the device AJ12Letter. See HPSLET.HLP for documentation.

Updated: 3 September 81

@Marker(Make,HPSLetter)
@Style(Spacing 1,Spread 1,Indentation 3)
@Set(Page = 1)

@Define(Ends, NoFill, LeftMargin 0, RightMargin 0, 
        Break, Spaces Kept, BlankLines Kept)
@Define(Body, Fill, LeftMargin 0, RightMargin 0, 
        Break, Above 1, Below 1, 
        Spaces Compact, BlankLines Break)
@Define(Annotations, NoFill, LeftMargin 0, Break, BlankLines Kept, 
        RightMargin 0, Spaces Kept, fixed -4)
@Equate(Annotation=Annotations)
   .fine(PS=Body, Above 1, Below 1)
   @define(Bye=Ends, LeftMargin 4inches, Above 2)

@String(Ext=')
@TextForm(Tope=

   @Parm(Ext,default=@Value(Ext))

   @Parm(Date,default=’@Value(Date)’)

"
)
@TextForm(Dear = 
   Dear @Parm(text),
   @Begin(body, EofOK)
"
)

@TextForm(Sincerely=
   @Begin(Bye)
   Yours Sincerely,

   @Parm(text)
@End(Bye)
"
)
@Begin(Text, Font CharDef, FaceCode R, EofOK)
@Begin(Ends, EofOK)
Lawrence's version of HPSLET.MAK

Updated: 3 September 81

@marker(Make,MyHPSLetter)
@strings(Ext='6296')

@comment[ Stuff from HPSLET ]

@Style(Spacing 1,Spread 1,Indentation 3)
@Set(Page = 1)

@Define(Ends, Nofill, LeftMargin 0, RightMargin 0, Break, Spaces Kept, BlankLines Kept)
@Define(Body, Fill, LeftMargin 0, RightMargin 0, Break, Above 1, Below 1, Spaces Compact, BlankLines Break)
@Define(Annotations, Nofill, LeftMargin 0, Break, BlankLines Kept, RightMargin 0, Spaces Kept, fixed -4)
@equate(Annotation=Annotations)
@Define(PS=Body, Above 1, Below 1)
@Define(Bye=Ends, LeftMargin 4inches, Above 2)

@TextForm(Top=

    @parm(Ext, default=@Value(Ext))
    @parm(Date, default='@Value(Date)')

"
)
@TextForm(Dear =
    Dear @Parm(text),
    @Begin(body, EofOK)

    "Textform(Sincerely= 
    "@Begin(Bye)
    Yours Sincerely, 

    @Parm(text)
    @End(Bye)
"
)

@comment[ new stuff ]

@Textform(MySincerely="@Sincerely(Lawrence Byrd)")
@Textform(Lawrence="@Sincerely(Lawrence Byrd)")

@Begin(Text, Font CharDef, FaceCode R, EofOK)
@Begin(Ends, EofOK)
@Top()
Device definition for the AJ using a 12 point pitch mounted with HPS large size headed paper. This is really only an adjusted LPT definition so it is not clever about the AJ in any way.

Most environments have been fixed to Above 1 and Below 1 as well.

Updated: 3 September 81

}
How to get complete (nice) listings of a BIB database

@make{text)
@device{lpt)
@style{Bibselect=Complete, References=Magic)
@Use{Bibliography="papers.bib")
@center{@b{
THE MECHO PROJECT
Research Papers and Working Papers
@}]
@bibliography

Use my ref. format (Pro)

a heading
Bibliography format definition for standard open-format alphabetic references.

@Style(Citations=4, Bibsequence=numeric, StringsMax=2000)
@Define(L1, LeftMargin 15, Indent -15, Above 1, Break)
@Define(L2, LeftMargin 18, Indent -3, Above 0, Break)

@LibraryFile(Standard)
@Enter(Text, Spacing 1, Spread 0, Spaces Tab, Justification off, Fill)
@Process(Bibliography)
@Leave(Text)
NAME

CD - Change Directory

SYNOPSIS

cd Destination

DESCRIPTION

CD allows you to move around easily between different directories and SFD's (sub file directories). If it can find the destination it will change both your default path and your default (file) protection. Normally you only have to give an SFD name, for example the command:

```
cd papers
```

(NB This mode of using CD may not work immediately for you. If not, then see the file CD.NEWS[400,444]). This command will change your path to [,,papers] if you have PAPERS,SFD in your directory. However, not only does CD notice SFD's straight in front of you (in your current path) it will also search around for the Destination. For example having moved to [,,papers] the command 'cd prolos' will move you (change your path) to [,,prolos] if you have PROLOG,SFD in your top directory (UFD). In general CD will try the following (in the given order):

1. Look in SWITCH.INI[,] to see if Destination is defined to be some arbitrary path. (This enables you to set up useful mnemonic names for interesting places).

2. Look to see if the appropriate SFD can be found as a daughter, sister or aunt to the current path (i.e. looking all the way up to the UFD).

3. Look for the SFD in your own top level directory, if that has not already been done (i.e. you may be in some other directory completely and wish to return).

4. Look to see if there is a similar 'logical path name' defined (i.e. one with the same name) and use the (first) path associated with that. (See the PATH command for details on logical path names and how to set them up. They enable one to refer to arbitrary paths like devices, and this meshes nicely with CD's ability to move you there. Note that they are different from a variety of confusingly similar TOPS10 'features' such as logical ASSIGNED names and ersatz devices. CD extension to ersatz device names is being considered.)

If this search fails then CD will leave you where you started. The full details of this search strategy do not need to be remembered; most of the time you will find that CD does exactly what you want. CD always prints a final message telling you where you have ended up, as well as showing other information, such as your new default protection (see below). In addition CD will also accept full path expressions (TOPS10 standard), e.g.:
The usual abbreviations may be used and the square brackets are optional (they were not used above). The following conveniences are also available:

- **cd**; Home (your top level directory)
- **cd -**; Current path
- **cd +**; Top level directory of current path
- **cd ~**; Up one SFD from current path

Thus the command "cd" on its own will always return you to your own directory. The command "cd -" is useful for seeing where you currently are.

CD will also change your default file protection every time you move. The need for this stems from TOPS10's primitive notions of file ownership. If you create a file in a directory of another PPN then the file effectively ends up owned by that PPN; the Self field now refers to him and the Group or Others field to you! You will not be able to read/edit/delete the file unless the Group or Others protection field allows it. Most people's normal default protection does not allow Others to do all these things. Thus you will be unable to manipulate files you yourself have created. To maintain control of files you create it is necessary to create them with more lax protections if you are not in your own directory. These circumstances, and inconveniences, are very common within any group of collaborating users. CD solves these problems by maintaining your default protection for you. If you do not have an explicit CD protection specification in your SWITCH.INI then CD uses a default strategy, which involves looking at your 'login/defprot' switch if you have one, and works as follows:

- **Given login/defprot:xyz**
- **Otherwise**

<table>
<thead>
<tr>
<th>(moved to)</th>
<th>Self</th>
<th>Group</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Self</td>
<td>xyz</td>
<td>00z</td>
<td>000</td>
</tr>
<tr>
<td>Group</td>
<td>055</td>
<td>005</td>
<td>000</td>
</tr>
<tr>
<td>Others</td>
<td>000</td>
<td>005</td>
<td>000</td>
</tr>
</tbody>
</table>

These labels have the obvious interpretation of destination paths which share your programmer number (Self), or project number (Group), or neither (Others). This strategy may seem a bit complicated, but in practice you will find that it means that CD will behave sensibly and the whole business of protections can be forgotten and left in its care. The login/defprot switch is a standard login switch (badly documented however) which acts at login time to set your jobs initial default protection. An example would be:

```
login /defprot:055
```

It is a good idea to have one of these in your SWITCH.INI, even if you wish to keep the overly paranoid current system default (which is 057). This is because CD's default strategy if this is missing will not keep your protection at 057.

Note that it is always possible to override all this default activity by an explicit specification in your SWITCH.INI, and in addition to setting up your protection changing strategy, your SWITCH.INI can also be used for defining names for useful destinations (as mentioned
above), and giving default settings for various switches. SWITCH.INI specifications are quite important for convenient use of CD among collaborating users. The best way to understand what these SWITCH.INI entries should look like is to examine someone else's SWITCH.INI file which uses CD definitions (a reference is given below). However a complete specification of CD's operation, and use of SWITCH.INI, is given in DOC:CD.DOC (but this is rather hard work).

SEE ALSO

DSKA:SWITCH.INI[400,441] - for examples of CD definitions
DSKA:CD.NEWS[400,444] - for recent news about CD (read this!)
DOC:CD.DOC - CD's specification
DEC10 Commands manual - especially the Introduction sect. 1.4.2
SETSRC program - a painful way of changing paths
PATH command - for manipulating "logical path names", it also changes paths (but is stupid about it).
DEC10 Monitor calls manual - to see how awful it all is

AUTHORS

Lawrence Byrd and Richard O'Keefe
Department of Artificial Intelligence
Hope Park Square
Edinburgh

Bugs, suggestions and monies to [400,441] (on ERCC DEC10) please.

NOTE

The specification, program and documentation are
Copyright Lawrence Byrd and Richard O'Keefe 1981
This file will contain odd bits of news about CD. However it is not really our intention to make any changes to the current version.

News 1

CD has now been placed in SYS:, however we have been unable to persuade the system staff to make it a full monitor command. For many of you the method of use documented will not work. I.e:

```
$ cd foo
?cd?
```

Just produces the error message as shown.

For immediate use you will therefore have to use the following, less convenient, method:

```
$r cd; foo
```

However, there is a fix to this problem. The systems staff have incorporated a slight change in the TOPS10 command scanner which allows programs which aren't proper monitor commands to be run as if they were. What happens is that when you type:

```
$ hello
```

the system realises that it isn't a command and it tries to do "$r sys:hello" instead. This means that any program in SYS: will effectively become a command. In particular, of course, it means that CD can be used as if it was a command - which is how it should be used.

In order for this to work for you must have a certain privilise bit set for your PPN. This can only be done by contacting User Support (Jeff Phillips in particular) and asking them to fix this up for you. This facility seems to be going by the name of 'Magic Bits' at the moment (so that's what to ask for). If you are a member of a group of users then it is undoubtedly sensible to ask for 'Magic Bits' for all the PPN's of your group at the same time.

I apologise for the additional hassle involved in using CD properly but the matter is beyond my control.

News 2

This is not about CD, but rather about the consequences of setting Magic Bits set up for your PPN. You should only bother to read this if you are interested in squeezing the maximum user-assistance out of TOPS10. In the new 7.01 monitor there is a facility called 'logical names' which is a (hacked up) add on to device names, ersatz devices and so forth. The principle way of using them is through the PATH command. For example:

```
$PATH stuff: = [400,441,bin]
```

sets up STUFF: as a logical name which stands for the disk area [400,441,bin]. Most of the file system will treat this name just like a device, for example in commands like:
.ty stuff: PPNS

; types the file PPNS[400,441,bin]

This will also work inside programs, editors and so forth. Note also that:

.cd stuff

will work, moving you to [400,441,bin] (this feature is documented in HLP:CD.HLP. Note that no colon should be used with the name stuff in this case).

The really interesting gimmick is that you can a) redefine SYS; to point to arbitrary bits of the filing system; b) redefine your library path in the same way; and c) specify that either or both of these involve LISTS of many different paths.

The combination of Magic Bits (all programs in SYS; are commands), logical names (SYS; can point anywhere, as your libraries), and CD; produces a very powerful way of using the system.

.example the following commands:

.path sys; = dska:[400,444], dske:[1,5], dske:[1,4]
.path lib;/search = dska:[400,441,bin], dska:[400,444]

would allow me to run programs from my library ([400,444]) and from normal SYS; (the two dske; places are the normal new; and sys;), all as if they were commands (given Magic Bits). Files in both [400,441,bin] and [400,444] would be available as library files, ie I wouldn't have to specify any directory to read them etc.

We find these facilities incredibly useful, and think that you probably will too. For further details consult the followings (I do not have time to write the missing documentation):

1) Ask User Support for a copy of the documentation for the PATH command (only they have the new 7.01 manuals),

2) Take a look at the files:

READ,ME[400,444] ; Top of the tree for docm on our

library area. Following this will ; give some idea of how we use it.

PATHS,HLP[400,444] ; Explains how to set up path ; definitions at login.

------------------------------------------------------------------------------------------------------------------

If you have questions concerning CD, or if you would like a copy to take it elsewhere, then send me a message via:

.tell byrd

<mensaje>

^z

Good luck
Lawrence
This file gives a detailed specification. For user documentation see the file -
HLP:CD.HLP

or just type "help cd" ]

Specification for the "CD" program.
=======================================

This program was originally intended to replace a primitive MIC macro which changed paths in a simple fashion. Its aim is to provide a convenient way of moving from one SFD to another, or indeed to any other PPN or path. It should also handle automatic default protection changes and defined names on a per user basis (using switch.ini).

The main features of the program are as follows:

1) The program is intelligent about finding SFDs. This involves looking in SWITCH.INI[ ], searching through daughter, sister and aunt SFDs, and trying the user's PPN (if the current path is not rooted there already), checking out logical path names etc. This will enable the use of just a simple (mnemonic) name for most of the destinations a user will normally want to get to.

2) Changes of the job's default protection (for files) are handled in a reasonable way. When moving to a path which is not under one's own PPN, it is necessary to "free" ones protection so that files created there can be deleted. Specification of the appropriate protection is via SWITCH.INI[ ].

3) The program ALWAYS reports on the final destination and the new default protection (whether they have changed or not). If any problems are encountered, the user's path and protection are left as they are, so that the user can examine the problem and try again. Reasonably specific error messages should help pin down the problem.

This specification was originally formed before the PATH command was available, however we had the documentation for it and it was decided that the facilities of CD were much to be preferred. A conscious design decision was that CD should provide a powerful, conceptually coherent, model for moving around the file system. It should not try to offer all the facets of the PATH UUO (a la SETSRC, PATH) since we feel that that offers a very poor conceptual model. Far too many different things are bundled into it. CD does one job, but does it properly.

Note: This specification has been updated to be a reasonably accurate description of the program as written. Hence (and regrettably) it is slightly more complicated than the original.

How to use CD.
-------------

The program can be run by one of:
.

  cd <input>  - if command
depending on where the program is and how it is set up. We shall henceforth assume that CD is set up as a command for that is by far the most effective way of using it. The form of <input> is the same regardless of which method is used. CD will also work (scanning the new command line) if it is started, continued or reentered; however this is unlikely to be of much importance if CD is set up as a command.

The <input> can be of the following form:

(All grammars are in a BNF like formalism where non-terminals are in angle brackets, and terminals are between single quotes, |:' stands for disjunction and concatenation is represented by spaces between items. Braces ( {...} ) indicate that some component may be repeated (>0). All terminals have been written in lower case, but CD actually regards upper and lower case as identical in every circumstance.

<input> ::= <path spec> <protection> <switches>
<path spec> ::= <lbrak> <base and trail> <rbrak>
<lbrak> ::= ’[’ | <empty>
rbrak> ::= ’]’ | <empty>
<base and trail> ::= <base> <trail>
  | <empty>
<trail> ::= {,’ <sfd name> }
<base> ::= <explicit base> | <implicit base>
<explicit base> ::= <project #> ’,’ <programmer #>
  | ’~’
  | ’+’
  | ‘~’
<implicit base> ::= <name>

<project #> ::= <<octal digit>}
<programmer #> ::= <<octal digit>}

Either or both parts of the PPN may be left out. As usual, they will default to the corresponding part of your Job's PPN. E.g., if your PPN is 123,456 then "cd 777" will be equivalent to "cd 123,777".

<sfd name> ::= <identifier>
<name> ::= <identifier>

<identifier> ::= <<letter> | <digit> | % | $ | <numeric>}
<numeric> ::= ’#’ <<octal digit>}

An identifier can consist of any number of characters, but will be truncated to six. You can include any sixbit character by siving its octal equivalent,
<protection> ::= '('<octal digit>')' 
|  <empty>

The protection is a standard DEC 3-digit protection code. The angle brackets may NOT be left out. Note that you don't need to specify this if you are happy with what CD gives you. Since you are shown what your new path and protection are, you can say 'cd <xyz>' if you want something else.

<switches> ::= {'<switch>'}
|  <empty>

<switch> ::= 'help'
|  'help:no'  - normal default
|  'help:yes'
|  'help:all'
|  'dir'
|  'dir:no'  - normal default
|  'dir:yes'
|  'scan'
|  'scan:no'  - normal default
|  'scan:yes'
|  'scan:asis'

The short form of these switches implies :yes, e.g., /dir means /dir:yes.

There is a flaw in the program which means that you have to use the long form if you have a setting in your SWITCH.INI which you want to override.

The help switch asks for the file HLP:CD.HLP to be printed out. If /help:all is used the file DOC:CD.DOC is printed instead (which is this file).

The dir switch specifies that a fast directory listing is to be performed after moving to the new path. It will happen even if there was an error.

The scan switch specifies whether or not SCAN will be set (see the system manuals about this). /scan:asis specifies that SCAN should be left set to whatever it was previously. Note that the default is /scan:YES.

START
IF the <input> does not conform to this syntax, an error message explaining the symptom is written, often indicating the culprit.

CD will then display your current path and protection, and exit leaving them unchanged.

IF the input is valid then the following occurs:

The actions specified by the switches are taken at some suitable point (either before or after the following).

IF <base and trail> is <empty>
    THEN Change the Job's path to the user's UFD (cannot fail)
ELSEIF <base> is an <explicit base>
    THEN Append the path specified by <base> to the <trail> and try to change the Job's path to this path. If unsuccessful provide an error message and leave the path as it was originally.

The <explicit bases> have the following interpretations:
<Project *>, <Programmer *> - stands for itself
'-' - stands for the current path
'+' - stands for the root (PPN part)
',' - stands for the current path with the last SFD removed (if there is one)

ELSEIF the <base> is an <implicit base> (ie a <name>)
THEN Look in SWITCH.INI[] for a definition of the <name>,
IF <name> is defined use the definition as an <explicit base> and
proceed as above.

OTHERWISE
Append the current path, <name>, and the <trail> together and try
to change the Job's path to this path,
IF this fails then reduce the current path by one SFD and try the
previous step again.
Do this until no more SFD's can be taken from the current path.
IF still unsuccessful, and if the root (UFD) of the current path is
different from the user's PPN then append the user's PPN, <name>,
and the <trail> together and try to change the Job's path to
this path (similar to original step).

OTHERWISE
Use the PATH UUO to see if the user has used <name> to define a
'logical path name'. (The names must be the same, the monitor level
syntax of using colon is irrelevant here). If such a logical path
name is defined then use the FIRST associated path as an
<explicit base> and proceed as above. (If there is more than one
associated path then a warning message is given but CD proceeds,
using the first associated path).

OTHERWISE
Give up. Provide an error message and leave the Job's path and
protection as they were originally.

CLOSE

COULD any appending of paths result in a path with more SFD's than the
allowed maximum (currently 5) this is an error - Provide an error
message and leave the Job's path and protection as they were
originally.

SHOULD the search be successful (ie Job's path is changed - even if this sets
it to what it was originally)
THEN reset the Job's default file protection:
  IF the command line had an explicit protection <Prot>, set the Job's
default protection to that.

OTHERWISE
Define the fields (SELF, GROUP, OTHERS) by scanning the user's
SWITCH.INI file using the following defaults for any field not
so defined:

  SELF  = xyz
  GROUP = 00z
  OTHERS = 000

where xyz are taken from the 'login/defprot:xyz' entry in SWITCH.INI.
If there is no login/defprot entry then xyz = 055.
An *asis* specification in SWITCH.INI will set all three fields to
the value of the Job's current default protection.
Use the final default path (as set or left, see above) to determine
the applicable field. If the programmer part of the path is the same
as the user's, use SELF (Note: it is NOT necessary for the project
part to also agree in this case). Otherwise, if the project part of
the path is the same as the user's, use GROUP, Otherwise use OTHERS.
Set the Job's default file protection to the value of the selected
field.

END

FINALLY Output a message to the user showing the final path and default
file protection (guarantee values by re-asking the monitor).
If the scan switch is not set then indicate this in the message.
If logical name processing was used then indicate this in the message.

FINISH

It should be noted that the <base and trail> system described here is more
tensive than what is described in the user documentation (HFL:CD.HLP).
...the HLP file only "cd foo" forms are mentioned. However, as can be seen,
forms like "cd foo,baz" or "cd foo,abc,xyz" also have well defined meanings
even when "foo" (the <base>) undergoes SWITCH.INI or logical path name
processing.

Examples
-----------

.cd 123,456,Pit

    Should set the path to [123,456,Pit]

.cd -

    Should leave the default path as it is (however the
full current path will be printed out)

.cd <155>

    If the current path is [555,444,prolos,code,notes] it will
be set to [555,444,prolos,code] and the protection to 155.

.cd foo

    If we assume that the user is [123,456] and that his
current path is [555,444,prolos,code], and that he has a
logical path name set up as if by:
    .path foo:=[111,246,feefie]
then providing that foo is not defined in switch.ini[123,456]
the following search would ensue:

    [555,444,prolos,code,foo]
    [555,444,prolos,foo]
    [555,444,foo]
    [123,456,foo]
    [111,246,feefie]

.cd foo,one,two
This example exhibits the <trail> mechanism. Using the assumptions used above the following search would ensue:

```
[555,444,prolos,code,foo,one,two]
[555,444,prolos,foo,one,two]
[555,444,foo,one,two]
[123,456,foo,one,two]
[111,246,feefie,one,two]
```

The SWITCH.INI file
---------------------

The user's SWITCH.INI file may contain lines providing information to CD. This may consist of <name> definitions, default protection settings, and default switch settings. Note that the SWITCH.INI file read will be SWITCH.INI[,] not any file of the same name in the current path (or whatever), is standard across TOPS10 programs which use SWITCH.INI.

SWITCH.INI entries can be as follows:

```
cd /Protection:( <Protdefs> )
cd /name:( <namedefs> )
cd /switches:( <switchdefs> )
```

where:

```
<Protdefs> ::= <Protdef>
   | <Protdef> ',' <Protdefs>
   | 'as is'.

<Protdef> ::= 'self' '=' <Prot>
   | 'group' '=' <Prot>
   | 'others' '=' <Prot>.

<Prot> ::= <octal digit>}

<namedefs> ::= <namedef>
   | <namedef> ',' <namedefs>.

<namedef> ::= <name> '=' <Pathspec2>.

<Pathspec2> ::= ( As <Pathspec> except that both the square brackets must be present to avoid ambiguity (commas separate <namedef>s) )

<switchdefs> ::= <switch> '{',' <switch>
```

Note that the (three digit) protection codes in a /Protection entry must NOT have angle brackets round them.

The SWITCH.INI file is read (if it exists) before anything else is done, and all of it is read. Two scans are performed: first looking for login entries, and secondly looking for cd entries. The login entries are looked at in an attempt to pick up the 'login/defprot' switch which, if present, is used
as a template in the default protection changing strategy (see above).
Note that this default may be then overridden by a proper cd/protection entry,
however we recommend the use of a login/defprot switch in your SWITCH.INI
(the default strategy based on this is maximally sensible).
[ This method of looking at other programs entries (ie login) is a more recent
addition and is not an idea we are completely happy with, It does happen to
be very convenient in this case. ]

If any SWITCH.INI line cannot be properly parsed
then an error message will be printed and the program will
exit. The simplest way of continuing is to temporarily rename the file.
The best way of continuing is to fix the file!
Note that it is not necessary for SWITCH.INI to exist for CD to work.

If there are repeated definitions for the same <name> or protection field then
only the first definition will be used, any others being ignored.

Here are some example entries taken from an imaginary SWITCH.INI:

    cd/switches! (scan, dir:no) ; these are the defaults anyway
    cd/protection! (self=005, group=005, others=000) ; a login/defprot:005
                                                 ; entry would do this
    cd/name! (Justine=[777,248])
    cd/name! (boadicea=[111,111])
    cd/name! (util=[123,456,util])
    cd/name! (flush=[123,456,blng,hole,y2])
    cd/name! (pros=[123,455,top,code])
    cd/name! (fred=[,457])

END SPEC
-------

SEE ALSO

HLP:CD.HLP - Help file, This provides user oriented documentation,
as well as other references.

AUTHORS

Lawrence Byrd Dept. of Artificial Intelligence
Hope Park Square Edinburgh

ERCC DEC10: [400,441] (Bugs/comments to here please)

Richard O’Keefe Dept. of Artificial Intellience
Hope Park Square Edinburgh

ERCC DEC10: [400,422]

NOTE

The specification, program and documentation are
Copyright Lawrence Byrd and Richard O’Keefe 1981
<table>
<thead>
<tr>
<th>Name</th>
<th>Extension</th>
<th>Len</th>
<th>Prot</th>
<th>Access</th>
<th>---</th>
<th>Creation ---</th>
<th>Mode</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSKA</td>
<td>[400,421,CD]</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>HLP</td>
<td>15</td>
<td>&lt;005&gt;</td>
<td>9-Jul-81</td>
<td>18:50</td>
<td>8-Jul-81</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>DOC</td>
<td>30</td>
<td>&lt;005&gt;</td>
<td>11-Jun-81</td>
<td>2:25</td>
<td>11-Jun-81</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>BLI</td>
<td>100</td>
<td>&lt;005&gt;</td>
<td>1-Jul-81</td>
<td>19:56</td>
<td>18-Jun-81</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>MSS</td>
<td>15</td>
<td>&lt;005&gt;</td>
<td>8-Jul-81</td>
<td>18:50</td>
<td>8-Jul-81</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>READ</td>
<td>ME</td>
<td>5</td>
<td>&lt;005&gt;</td>
<td>9-Jul-81</td>
<td>13:39</td>
<td>9-Jul-81</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>CCL</td>
<td>5</td>
<td>&lt;005&gt;</td>
<td>9-Jul-81</td>
<td>13:50</td>
<td>9-Jul-81</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Total of 170 blocks in 6 files on DSKA: [400,421,CD]
;; CD.CCL : All the bits of the CD program etc.
;;
;; This is set up as a CCL file for use with BACKUP. Use by typing
;; "@cd[400,421,cd]" to BACKUP in order to copy stuff to tape.
;; \* NB Not yet tested!! (hope the comments are OK...,)
;;
ssname 'CD'
save  dska:cd.ccl[400,421,cd],- ; This file
       dska:cd.bli[400,421,cd],- ; The (BLIS10) source code
       sys:cd.exe,- ; The current EXE
       dska:cd.doc[400,421,cd],- ; The specification
       dska:cd.mss[400,421,cd],- ; The (SCRIBE) source for cd.hlp
       dska:cd.hlp[400,421,cd],- ; The user help file
       dska:cd.new[400,444] ; Latest CD news
Program  CD.BLI

Author  R. A. O'Keefe

Date   9-13 September 1980
Updates  20 March 1981 - odd pruning
          14 May 1981 - logical names
          20 May 1981 - various rationalisations
          25 May 1981 - read LOGIN/DEFPRT

Site    Department of Artificial Intelligence, Edinburgh.

Purpose one-line SetSRC to simplify changing default PATH
NB this antedates PATH, and provides other features

See    CD.HLP - User documentation
        CD.DOC - Specification

Copyright R. A. O'Keefe 1981

% module cd(stack, fsave, timer=external(six12), debug) =

% module cd(stack, lowses) =

begin
  bind ProgramName = sixbit 'CD';

  bind No = 0;    ! false logical value, /switch:NO or /NOSwitch
  bind Yes = 1;   ! true logical value, /switch:YES or /switch
  bind EOS = "?O"; ! End Of Strings character
  bind cr = "?M"; ! Carriage Return
  bind lf = "?J"; ! Line Feed
  bind tab = "?I"; ! like a space

  own  ch;      ! character which terminated previous token
  own  TextPtr; ! points to next character of command/switch line
  own LineBuf[30]; ! holds current command/switch line
  own SixBuf[5];  ! holds sixbit identifiers
  own ErrorCount; ! errors detected in *this* run

All terminal I/O is done using TTCALLs, rather than opening a
channel on TTY: . The main reason for this is that I learned about
TTCALLs first, and feel happier with them. Also, I need to rescan
the command line, so TTCALLs are needed for input anyway.

putasc(ch) types a single right-justified ASCII character ch.
putext(st) types the Ascii strings literal st (use single quotes)
putsix(id) types a sixbit identifier id without trailing blanks.
putoct(nb) types the (unsigned) number nb in octal.
puterror(t,n) is rather kludgy, and relies on knowing the globals.
GetTermLine(cmd,buf) reads a line, special handling for commands.
machop ttcall = #051;
machop CallI = #047;

macro putasc(ch) =
  (vres = ch; ttcall(1, vres)) $;

macro putext(st) =
  ttcall(3, uplit ascii st) $;

routine putsix(SixChars) =
  while .SixChars neq 0 do begin
    vres = .SixChars<30, 6>+32;
    if .vres sea "A" and .vres leq "Z" then vres = .vres+32;
    putasc(.vres);
    SixChars = .SixChars ~ 6
  end;

routine putoct(oct) =
  begin
    register w, n;
    w = .oct~18;
    n = 0;
    do (n = n+1; w = w^(3)) until .w<18, 18> eal 0;
    do (putasc(.w<15, 3>+"0") ; w = .w^(3)) until (n = n-1) eal 0;
  end;

routine PutError(ErrorType, ErrorNumber) =
  begin
    bind Messages = uplit(
        % 0% uplit ascii 'unknown switch',
        % 1% uplit ascii 'unknown attribute',
        % 2% uplit ascii 'unknown value',
        % 3% uplit ascii 'missing switch name',
        % 4% uplit ascii 'missing attribute name',
        % 5% uplit ascii 'missing , }, or }',
        % 6% uplit ascii 'missing " or ",
        % 7% uplit ascii 'Nothing should Precede [',
        % 8% uplit ascii 'Nothing should come between [ and ]',
        % 9% uplit ascii 'Protection or switches expected here',
        %10% uplit ascii 'missing sfd name',
        %11% uplit ascii 'meaningless after ]',
        %12% uplit ascii 'meaningless after <protection>',
        %13% uplit ascii 'Protection code should be followed by >',
        %14% uplit ascii 'unexpected text',
        %15% uplit ascii 'SWITCH.INI too long, stopping before?M?J',
        %16% uplit ascii 'Too many SFDs; the limit is 5',
        %17% uplit ascii 'no changes made',
        %18% uplit ascii '+ or - should be followed by , or ]',
        %19% uplit ascii 'Device has several paths; first taken',
        %20% uplit ascii 'Badly formed number',
        %21% uplit ascii 'Sorry, can't read the directory',
        %22% uplit ascii 'Sorry, can't find it',
        %23% uplit ascii 'Sorry, can’t change protection',
        %24% uplit ascii 'Brackets don’t match',
        0)
    putasc(if .ErrorType str 2 then "%" else "??")
    putasc(" ");
    if .ErrorType str 2 then ErrorCount = .ErrorCount-1;  ! warnings only
case .ErrorType mod 3 of set
  %0%  begin
    ttcall(3, .Messages[.ErrorNumber]);
  end;
  %1%  begin
    putsix(.SixBuf);
    putext(' - ');
    ttcall(3, .Messages[.ErrorNumber]);
  end;
  %2%  begin
    register Eptr;
    ttcall(3, .Messages[.ErrorNumber]);
    putext(' - ');
    Eptr = .TextPtr;
    putasc(scann(Eptr));
    while .vres neg EOS do putasc(scani(Eptr)) end;
    tes;
    putasc(cr); putasc(lf);
  ErrorCount = .ErrorCount+1;
  end;

routine GetTermLine(Command, Buffer) =
  begin
    register char, ptr, pts;
    if .Command then begin
      ifskip ttcall(8, 1) then begin
        putasc("*"); ! can't rescan command line, ask user for another
        Command = No; ! which won't need special command handling,
        ! otherwise rescan done, special handling needed.
      end;
      ptr = (.Buffer)<36,7>;!
    while 1 do begin
      ttcall(4, char); ! read a single character, but wait for a whole line
      if char eq " " then begin
        replacei(ptr, .char);
      end else
      if #01400016200 ^ (-,char) then begin
        exitloop; ! break char: bel,lf,vt,ff,~z,esc
      end else
      if .char neg cr and .char neg EOS then begin
        replacei(ptr, .char);
      end end if %
    end % while %
    replacei(ptr, EOS); ! AsciZ string terminator
    replacei(ptr, EOS); ! needed for empty command lines.
    replacei(ptr, EOS); ! just to be safe.
    ptr = (.Buffer)<36,7>;
    if .Command then begin
      do char = scani(ptr) until .char neg " " and .char neg tab;
      ptr = SixBuf<36,6>;
      SixBuf[0] = sixbit ' '; ! command name
    while 1 do begin
      vres = if .char str "." then .char-64 else .char-32;
      if .vres less sixbit 'A' then exitloop;
      if .vres str sixbit 'Z' then exitloop;
    end % while %
    putext(' - ');
    putasc(scani(ptr));
  end;
replacei(ptr, 'vreg');
    char = scani(ptr);
end;
if .SixBuf neq ProgramName then begin
  % look for a comment character %
  while .char neq ';' and .char neq '!' and .char neq '"' and .char neq EOS do char = scani(ptr);
end;
end;
return .ptr
end;

====================================================================
!
! CD uses disc input for two purposes: to read the SWITCH.INI file
! and to read a CD.HLP or CD.DOC file. As the use of these two files
! doesn't overlap, a single fixed channel is adequate. Also, CD is
! quite capable of doing without either of them, so fancy error
! detection and recovery isn't needed. Furthermore, the file names
! are fixed too, so passing the components to OpenFile as sixbit is
! not too much of a burden. The routine GetFileLine has a parameter
! "crlf" which is just to make GiveHelp prettier.
!
! GiveHelp is a fairly simple-minded routine which prints <fn>.DOC
! {if level >= 2} or <fn>.HLP {if level =1 or <fn>.DOC unavailable}
! or its third parameter {if level = 0 or <fn>* unavailable} unless
! that is zero. Now that the program has become a command, its .HLP
! file is HLP:CD.HLP and its .DOC file is DOC:CD.DOC. It is easy to
! put them wherever you want by editing GiveHelp, but let users know.
!
! FastDirectory is given a path block (see later on in this program
! for a description of path blocks) and lists all the files it can
! find in that directory. It searches all the devices in the Job's
! search list, so it may miss some. The output resembles DIR/FAST,
! but it is in lower case. It will not be enhanced further.
!
====================================================================

own FileNameBlock[4] = (4;0); ! name,extn,0,ppn
own BufferBlock[3] = (3;0);
own DeviceBlock[3] = (0, sixbit 'DSK', BufferBlock[0]<0,0>);

bind BufferPtr = BufferBlock[1]; ! points to next character in buffer
bind BufferCnt = BufferBlock[2]; ! counts characters left
bind Channel = 1; ! this channel is reserved for SWITCH.INI

macro CloseFile =
begin
  machop close = $070;
  close(Channel, DeviceBlock);
end $;

routine OpenFile(Device, Filnam, Ext, P=nn) =
begin
  machop open = $050, close = $070, lookup = $076;
  DeviceBlock[1] = .Device;
  FileNameBlock[0] = .Filnam;
FileNameBlock[1] = .Ext and (-1)^18;
FileNameBlock[2] = 0;
FileNameBlock[3] = .Pgn;
if_skip open(Chanell, DeviceBlock) then begin
  if_skip lookup(Chanell, FileNameBlock) then return Yes; % success %
  CloseFile; % there's a device but no file %
end;
return No
end;

routine GetFileLine(buffer, crlf) =
begin
  machop in = $056;
  register nchars, char, ptr;

  nchars = 0;
  ptr = (.buffer)<36,7>
while 1 do begin
  if (BufferCnt = BufferCnt-1) lea 0 then begin
    if_skip in(Chanell, 0) then return -1;
  end;
  char = scani(BufferPtr);
  if .char sea " " then begin
    replacei(ptr, .char);
    nchars = .nchars+1
  end else
  if $01400016200 ^ (-.char) then begin
    exitloop % a break character %
  end else
  if .char neq cr and .char neq EOS then begin
    replacei(ptr, .char);
    nchars = .nchars+1
  end;
end % while 1 %;
if .crlf then (replacei(ptr, cr); replacei(ptr, lf); nchars = .nchars+2);
replacei(ptr, EOS);
return .nchars
end;

routine GiveHelp(ProgName, Level, Extra) =
begin
  bind HelpDev = sixbit 'HLP'; ! These two paths can be anywhere
  bind HelpFPN = 0; ! but should be where people look
  bind DocuDev = sixbit 'DOC'; ! for them, 0 is just the proper
  bind DocuFPN = 0; ! default for ersatz devices.
  local HelpLine[30];

  if .Level sea 2 then begin
    if OpenFile(DocuDev, .ProgName, sixbit 'DOC', DocuFPN) then begin
      while GetFileLine(HelpLine, Yes) sea 0 do ttcall(3, HelpLine);
      CloseFile;
    end else begin
      Level = 1;
      end
  end % level 2 %;

  if .Level eq 1 then begin
    if OpenFile( helpDev, .ProgName, sixbit 'HLP', HelpFPN) then begin
      while GetFileLine(HelpLine, Yes) sea 0 do ttcall(3, HelpLine);
      CloseFile;
    end
  end;
end else begin
  Level = 0;
end
end % level 1 %;

if Level <= 0 then begin
  if Extra neq 0 then begin
    ttcall(3, (.Extra)<0,36>);
  end else begin
    ttcall(3, uplit asciiz 'Sorry, no help available. ?M?J');
  end
  end % level 0 %;
end;

routine FastDirectory(PathBlock) =
begin
  macho calli = $047, in = $056;
  bind JobStr = $047; ! calli to examine this Job's search list
  bind binarymode = $14, ascii_mode = $00;
  bind PB = ,PathBlock;
  local SomeSeen, NextDevice[3];

  DeviceBlock[0] = binarymode; ! HACK
  SomeSeen = 0; ! none of directory read yet
  NextDevice[0] = -1; ! start from the beginings
  while 1 do begin
    vres = 3*18 + NextDevice<0,0>;
    ifskip calli(vres, JobStr) then (%ok%) else exitloop; ! error
    if .NextDevice eal 0 then exitloop; ! _FENCE
    if .NextDevice eal-1 then exitloop; ! end of search list
    ! try to open the directory on that device
    if begin
      if .PB[3] eal 0 then begin ! no SFDs
        OpenFile(,NextDevice, ,PB[2], sixbit 'UFD', 1^18+1)
        ! DSKA:U000400 000422.UFD[1,1], for example
      end else
      if .PB[4] eal 0 then begin ! one SFD
        OpenFile(,NextDevice, ,PB[3], sixbit 'SFD', ,PB[2])
        ! DSKA:FILES.SFD[400,422], for example
      end else begin
        local LastSfd, LookupPath[10];
        LookupPath[0] = LookupPath[1] = 0;
        incr i from 2 to 9 do
          if (LookupPath[.i] = .PB[.i]) eal 0 then begin
            LastSfd = .LookupPath[.i-1];
            LookupPath[.i-1] = 0;
          exitloop;
          end
          OpenFile(,NextDevice, ,LastSfd, sixbit 'SFD', LookupPath<0,0>)
          ! DSKA:ISNONE.SFD[400,422,FILES], for example
        end
    end then begin
      SomeSeen = .SomeSeen+1;
      putasc(tab);
      putasc(tab);
      putsix(,NextDevice); putext('!?M?J');
      while 1 do begin
        if (BufferCnt = ,BufferCnt-2) eal 0 then
          ifskip in(Channel, 0) then exitloop;
        if scani(BufferPtr) eal 0 then exitloop;
      end
      putasc(tab);
      putasc(tab);
      putsix(,NextDevice); putext('!?M?J');
routine setsix(P7, %into% buffer) =
   begin
      register pt6, pt7, char;
      .buffer = 0;
      pt7 = ,< P7;  
      pt6 = ,< (buffer)<36,6>;
      while 1 do begin
         char = scani(pt7);
         if .char sea "0" and .char lea "9" then begin
            replacei(pt6, ,char-32)
         end else
         if .char sea "A" and .char lea "Z" or .char eal "%" or .char eal "$" then begin
            replacei(pt6, ,char-32)
         end else
         if .char sea "a" and .char lea "z" then begin
            replacei(pt6, ,char-64)
         end else
         if .char eal "$" then begin
            % pt6 points to 3-bit digits here %
            pt6<24,6> = 3;  ! byte-size = octal
            pt6<30,6> = .pt6<30,6>+3;  ! same top bit as before
            while (char = scani(pt7)) sea "0" and .char lea "7" do
               replacei(pt6, ,char-"0");
            if not .pt6<30,6> then replacei(pt6, 0);
            pt6<24,6> = 6;  ! byte-size = sixbit
            pt6<30,6> = .pt6<30,6>-3;  ! same top bit as before
         end else
         if .char eal "\" then begin
            char = scani(pt7);
            replacei(pt6, % sixbit value of char = %
                  if .char lea "\" then sixbit "" else
                  if .char str "-" then .char-64 else .char-32);
replacei(pt6, ,char)
end else
if .char neq " " and .char neq tab then
exitloop
end % while 1 %;
.p7 = .pt7;
return if .char eql ";" or .char eql "!" then EOS else .char
end;

routine setint(pt7, radix, answer) =
begin
resister pt7, char, number;
local NumSign;

pt7 = .,pt7;
NumSign = +1;
number = 0;
while 1 do begin
char = scani(pt7);
if .char sgeq "0" and .char leq "9" then begin
if (char = .char-"0") str .radix then PutError(2, 20);
number = .number*.radix + .char;
end else
if .char sgeq "A" and .char leq "Z" then begin
if (char = .char-("A"-10)) str .radix then PutError(2, 20);
number = .number*.radix - ("A"-10) + .char;
end else
if .char sgeq "a" and .char leq "z" then begin
if (char = .char-("a"-10)) str .radix then PutError(2, 20);
number = .number*.radix - ("a"-10) + .char;
end else
if .char eql "+" then begin
NumSign = +1;
end else
if .char eql "-" then begin
NumSign = -1;
end else
if .char neq " " and .char neq tab then
exitloop;
end % while %;
.p7 = .pt7; % look at it with scanN not scanI %
.answer = .number*.NumSign;
return if .char eql ";" or .char eql "!" then EOS else .char
end;

<<<
routine setFPN(pt7, FPN) =
begin
local pt, ch, FPN;

pt = pt = .,pt7;
do ch = scani(pt) until .ch neq " " and .ch neq tab;
if .ch neq "[" then pt = .,ps;
ch = setint(pt, 8, FPN);
if .PN neq 0 then (.FPN)<18,18> = .PN;
if .ch eql "]" then begin
  ch = setint(pt, 8, FPN);
  if .PN neq 0 then (.FPN)<0,18> = .PN;
end;
if .ch eql "]" then
do ch = scani(pt) until .ch ean * * and .ch neq tab;
return .ch
end;

>>%

routine setext(p7, buffer, nwords) =
! move a text from p7 to buffer, and pad it out with at least one EOS.
! nwords is set to the number of *words* moved. The delimiter is returned.
begin
register char, pt7, ptb;
local p$7, nchars, Quote, BracketCount, CloseBrackets[40];

pt7 = .p$7;
ptb = (.buffer)<36, 7>;

nchars = 0;   ! no characters moved yet
Quote = 0;    ! not inside a quoted string
BracketCount = 0;  ! no open brackets seen

while 1 do begin
char = scani(pt7);
if .char eae1 EOS then begin
exitloop
end else
if .char eae1 ,Quote then begin
$p$7 = .pt7;
if scani(pt7) eae1 .char then begin
replacei(ptb, .char);
nchars = .nchars+1;
end else begin
pt7 = .p$7;
Quote = 0;   ! strings finished
end
end else
if .Quote neq 0 then begin
replacei(ptb, .char);
nchars = .nchars+1;
end else
if .char eae1 ;* or .char eae1 !* then begin
exitloop;   ! comment
end else
if .char eae1 *, or .char eae1 /* then begin
if ,BracketCount eae1 0 then exitloop;
replacei(ptb, .char);
nchars = .nchars+1;
end else
if .char eae1 (( or .char eae1 [" or .char eae1 {" then begin
CloseBrackets[BracketCount] =
if .char eae1 *( then ")" else
if .char eae1 [" then "]" else ");
BracketCount = .BracketCount+1;
replacei(ptb, .char);
nchars = .nchars+1;
end else
if .char eae1 ")" or .char eae1 "]" or .char eae1 "}" then begin
BracketCount = .BracketCount-1;
if ,CloseBrackets[BracketCount] neq .char then begin
PutError(2, 24);  ! mis-matched brackets
while .BracketCount neq 0 do
if ,CloseBrackets[BracketCount = .BracketCount-1]
eae1 .char then exitloop;  ! this one, that is
if .CloseBrackets/.BracketCount/ neq .char then exitloop;
end % if mis-matched %;
replacei(ptb, .char);
nchars = .nchars+1;
end else
if .char eal """" or .char eal """" then begin
Quote = .char;
end else
if .char eal """" then begin
replacei(ptb, .char);
replacei(ptb, scani(pt7));
nchars = .nchars+2;
end else begin
replacei(ptb, .char);
nchars = .nchars+1;
end % if %;
end % while 1 %;
if .BracketCount neq 0 then PutError(2, 5);
if .Quote neq 0 then PutError(2, 6);
do replacei(ptb, EOS) until (nchars = .nchars+1) mod 5 eal 0;
.nwords = .nchars div 5;
.pt7 = .pt7;
return if .char eal ";;" or .char eal "!" then EOS else .char;
end;

routine ToOctal(SixChars) =
begin
register n, w;
n = 0;
w = .SixChars;
while .w neq 0 do begin
if .w<30<6> lss sixbit "0" or .w<30<6> str sixbit "7" then begin
SixBuf[0] = .SixChars;
PutError(1, 20);
return 0
end;
end;
.n = .n^3 + .w<30,3>;
w = .w^6;
end;
return .n;
end;

=====================================================================
!
!
To set switch settings from SWITCH.INI, you call ReadSwitches(Name)
where Name is the program name in sixbit, which is used to select
the lines of SWITCH.INI which will be inspected. These lines have
the form
  <progsname>/{<assignment>},...
where
  <assignment> = <attribute>!{<binding>{<binding>}},...
      | <attribute>!<binding>
      | <attribute> = <text>    -- login/name:"Fred"
      | <attribute>
  <attribute> = <identifier>    (a valid SIXBIT identifier)
  <binding> = <identifier> = <text>

The syntax for identifiers is slightly non-standard, namely

<identifier> = <non-special>...
<non-special> = A .. Z ! a .. z ! 0 .. 9 ! $ ! % ! # octaldigits!
\"any ASCII printing character\"
Lower-case letters are equivalent to upper case. \"!\" introduces one or more octal characters which are combined in pairs to form sixbit.
A quoted character ('X) stands for itself, e.g. '\'. Identifiers are converted to sixbit and truncated to six characters.

A text must be parenthesis balanced, but may be enclosed in quotes (either ' or ') in which case the contents may be unbalanced. The quotes are taken off before the text is stored as an ASCIZ strings.

ReadSwitches puts the results of its scan in the arrays AttrInfo and BindInfo. An entry in AttrInfo looks like this:

```
                __________________
               | ATTRNAME     |
               | TAIL         | HEAD
```

where AttrName is a sixbit attribute name, Head points to its first binding (0 if there are none), and Tail points to its last binding. The bindings themselves live in BindInfo and look like this:

```
                __________________
               | BINDNAME     |
               | KIND         | NEXT
               | TEXT         |
```

where BindName is the identifier of the binding, Next points to the next binding for the attribute, and Kind says whether there was (=1) or was not (=0) a text. If there was, it follows as an AsciiZ string NOT in sixbit. A string which will later be converted to sixbit will be handled by a routine which understands the same conventions as the one which converts identifiers.

I've had to add <attribute>=<text> to cope with login/ entries. I do this by hallucinating <attribute>!' VALUE=<text>, which can't be mistaken for the ordinary case. Drat that HT!

A pointer to the next slot for an attribute is kept in NextAttrPtr, and a pointer to the next slot for a binding in NextBindPtr. The finished attribute table will end with two successive zeros.

```
own MyJob, MyPPN; ! set by GetSwitchIniProtections
bind AttrSize = 30; ! suesestimate of number of distinct attributes
own AttrInfo[AttrSize*2]; ! see below
```
own NextAttrPtr;  ! points to next free word in AttrInfo
own ThisAttrPtr;  ! set by FindAttr, used by FindBind

macro AttrName(Ptr) = (.Ptr)[0] $;
macro head(Ptr) = (.Ptr)[1]<18,18> $;
macro tail(Ptr) = (.Ptr)[1]<0,18> $;

bind BindSize = 200$; ! This allows about 40 *name:*'s + all else
own BindInfo[BindSize*3];
own NextBindPtr;  ! points to next free word in BindInfo
own ThisBindPtr;  ! set by FindBind, used by SetXXX

macro BindName(Ptr) = (.Ptr)[0] $;
macro Kind(Ptr) = (.Ptr)[1]<18,18> $;
macro Next(Ptr) = (.Ptr)[1]<0,18> $;
macro Text(Ptr) = (.Ptr)[2]<36, 7> $;

bind Asis = 4$;  ! (sometimes) absent switch or /SWITCH:AS IS
bind All = 2$;  ! /switch:HELP=ALL

routine FindAttr(ThisAttr, PutItIn) =
! point ThisAttrPtr to the entry for ThisAttr, and return Yes.
! if there is none, put it in if PutItIn=Yes, otherwise return No,
begin
  decr TempAttrPtr from .NextAttrPtr-2 to AttrInfo do
    if .AttrName(TempAttrPtr) eal .ThisAttr then begin
      ThisAttrPtr = .TempAttrPtr;
      return Yes
    end;
if .PutItIn then begin
  ThisAttrPtr = .NextAttrPtr;
  NextAttrPtr = .NextAttrPtr+2;
  (.ThisAttrPtr)[0] = .ThisAttr;
  (.ThisAttrPtr)[1] = 0;
  return Yes
end else begin
  return No
end
end;

routine FindBind(ThisBind) =
! look through the bindings of ThisAttrPtr for one for .ThisBind,
! if one is found, point .ThisBindPtr to it and return Yes, otherwise No,
begin
  ThisBindPtr = .head(ThisAttrPtr);
  while .ThisBindPtr nea 0 do begin
    if .BindName(ThisBindPtr) eal .ThisBind then return Yes;
    ThisBindPtr = .next(ThisBindPtr)
  end;
  return No
end;

routine FindBoth(ThisAttr, ThisBind) =
  return if FindAttr(.ThisAttr, No) then FindBind(.ThisBind) else No;

routine SetSwitch(Name, Variable) =
if FindBind(.Name) then begin
  if .Kind(ThisBindPtr) then begin
    local ValuePtr, NextChar;
    ValuePtr = Text(ThisBindPtr);
    NextChar = setsix(ValuePtr, SixBuf);
    .Variable =
      if .SixBuf<30, 6> eal sixbit 'Y' then Yes else
      if .SixBuf<30, 6> eal sixbit 'N' then No else
      if .SixBuf<24,12> eal sixbit 'AS' then AsIs else
      if .SixBuf<24,12> eal sixbit 'AL' then All else
        (PutError(1, 2); ..Variable); % don't change it, wrong %
  end else begin
    .Variable = Yes;
  end;
  return Yes;
end else
if FindBind(sixbit 'NO''~24 + .Name'~(-12)) then begin
  .Variable = No;
  return Yes;
end else begin
  return No;
end;

routine SetNumber(Name, Variable, Radix, Default) =
  if FindBind(.Name) then begin
    if .Kind(ThisBindPtr) then begin
      local ValuePtr, NextChar;
      ValuePtr = Text(ThisBindPtr);
      NextChar = setint(ValuePtr, .Radix, .Variable);
    end else begin
      .Variable = .Default;
      return Yes;
    end;
end else begin
  return No;
end;

%<

routine SetPPN(Name, Variable, Default) =
  if FindBind(.Name) then begin
    if .Kind(ThisBindPtr) then begin
      local ValuePtr, NextChar;
      register FirstChr;
      ValuePtr = Text(ThisBindPtr);
      NextChar = setsix(ValuePtr, SixBuf);
      FirstChr = .SixBuf<30,6>;
      if .FirstChr eal sixbit 'N' then .Variable = 0 else
      if .FirstChr eal sixbit 'Y' then .Variable = .Default else
      if .FirstChr eal sixbit 'A' then (%% don't change %%) else
        begin ValuePtr = Text(ThisBindPtr);
          .Variable = .Default;
          NextChar = setPPN(ValuePtr, .Variable);
        end;
    end else begin
      .Variable = .Default;
    end;
    return Yes;
  end else
if FindBind(sixbit 'NO' + .Name~(-12)) then begin
  .Variable = 0;
end;
return Yes;
end else begin
  return No;
end;

>>%

routine SetStrings(Name, Variable, Default) =
  if FindBind(.Name) then begin
    if .Kind(ThisBindPtr) then begin
      .Variable = Text(ThisBindPtr);
    end else begin
      .Variable = .Default;
    end;
    return Yes;
  end else begin
    return No;
  end;

routine value(PutAtHead) =
  begin
    ThisBindPtr = .NextBindPtr;
    if .PutAtHead eal sixbit " VALUE" then begin ! HACK
      SixBuf = .PutAtHead;
      PutAtHead = No;
      ch = "="
    end else % HACK over %
    ch = setsix(TextPtr, SixBuf);
    if .SixBuf eal 0 then begin
      PutError(2, 4);
      SixBuf = sixbit " ERROR";
    end;
    (.ThisBindPtr)[0] = .SixBuf;
    (.ThisBindPtr)[1] = if .ch eal "= " or .ch eal "::" then 1^18 else 0;
    NextBindPtr = .NextBindPtr+2;
    if .ch eal ": " or .ch eal "=" then begin
      local Nwords;
      ch = setext(TextPtr, .NextBindPtr, Nwords);
      NextBindPtr = .NextBindPtr+.Nwords;
    end % if .ch eal "=" %;
    if .head(ThisAttrPtr) eal 0 then begin
      head(ThisAttrPtr) = .ThisBindPtr;
      tail(ThisAttrPtr) = .ThisBindPtr;
    end else begin
      if .PutAtHead then begin
        next(ThisBindPtr) = .head(ThisAttrPtr);
        head(ThisAttrPtr) = .ThisBindPtr;
      end else begin
        next(tail(ThisAttrPtr)) = .ThisBindPtr;
        tail(ThisAttrPtr) = .ThisBindPtr;
      end;
    end;

routine assignment =
  begin
    local TempPtr;
    ch = setsix(TextPtr, SixBuf);
    if .SixBuf eal 0 then begin
      PutError(2, 3);
      SixBuf = sixbit " ERROR";
end;
FindAttr( , SixBuf, Yes);
if ,ch eal "!" or ,ch eal "=" then begin
    TempPtr = .TextPtr;
    do ch = scani(TempPtr) until ,ch neq " " and ,ch neq tab;
    if ,ch eal "(" then begin
        TextPtr = ,TempPtr;
        do value(No) while ,ch eal ",,;";
        if ,ch neq ")" then PutError(2, 5);
        while ,ch neq " " and ,ch neq tab do ch = scani(TextPtr);
    end else
    if ,ch see "0" and ,ch leq "9" or ,ch see "A" and ,ch leq "Z"
or ,ch see "a" and ,ch leq "z" then begin
        value(No) ! single binding
        end else begin
            value(sixbit " VALUE")   ! HACK
        end
    end
end

routine ReadSwitches(ProsName) =
begin
    NextAttrPtr = AttrInfo[0];
    NextBindPtr = BindInfo[0];
    if OpenFile(sixbit 'DSK', sixbit 'SWITCH', sixbit 'INI', .MyPPN) then begin
        while GetFileLine(LineBuf, No) neq 0 do begin
            TextPtr = LineBuf<36,7>; ch = setsix(TextPtr, SixBuf);
            if SixBuf eal ProsName then
                if ,NextAttrPtr nea NextAttrPtr-2
                or ,NextBindPtr nea NextBindPtr-20 then begin
                    TextPtr = LineBuf<36,7>
                    PutError(2, 15);
                    exitLoop
                end else
                while ,ch eal "/" or ,ch eal ",," do assignment();
            end % while %;
            CloseFile;
        end % if %;
        (.NextAttrPtr)[0] = 0;
        (.NextAttrPtr)[1] = 0;
    end;

=== The following routine is just a debugging aid ===

%<<
routine PrintSwitches(ProsName) =
begin
    local AttrPtr, BindPtr;
    AttrPtr = AttrInfo[0];
    while ,.AttrPtr nea 0 do begin
        BindPtr = .head(AttrPtr);
        if ,BindPtr nea 0 then begin
            do begin
            end
A file protection is a nine-bit field \(<\text{Self \ Group \ Others}>\) where each field is an access code from 0 (least) to 7 (most) protection. CD keeps track of three complete default protections, 
\text{Self}, the protection when path's programmer number = yours 
\text{Group}, the protection when path's project number = yours 
\text{Other}, the protection to be used in other cases. 
Note that each of these is a full nine-bit protection field. 
Only the first assignment for each in SWITCH.INI has any effect. In addition, CD/PROTECTION\text{:ASIS} tells CD not to change your current protection at all. NB this is ONLY available in SWITCH.INI. You can specify a protection 8:xyz by putting \(<xyz>\) in the command, but there is no /protection switch in commands, only in SWITCH.INI.

Henry Thompson wants CD to pick up its protection without any help from him. The new scheme is that the default protection is taken from LOGIN/DEFFPROT:xyz if it exists, or is set to 055. Some code has been moved from ConsultMonitor into the new routine which does the new stuff, as it has to prepare for ReadSwitches. SWITCH.INI is now read exactly twice, however often CD is continued. I have taken advantage of this to look at logins/scan or logins/noscan as well, which is why it appears here and has no other business.

---

own Scan = Yes; ! should the /scan flag be set/reset/left alone?  
own Self = $055; ! the protection to use in your own area(s) 
own Group; ! the protection to use in your group's areas 
own Other = $000; ! the protection to use elsewhere. 
own Protection; ! the current/new default protection.
%<
routine PrintProtection(NewOld, Protection) =
begin
  putext('"Your "');
  putsix(.NewOld);
  putext('" default protection is "');
  putasc(.Protection<6>*"0");
  putasc(.Protection<3>*"0");
  putasc(.Protection<0>*"0");
  putext('"\?M?J\"");
end;
>%

routine GetProtectionDefault =
begin
  bind GetTab = #041, DefProt = #140;
  vres = .MyJob^18 + DefProt;
  return ifskirn calli(vres, GetTab) then .vres<27,9> else 0
end;

tine SetProtectionDefault(Protection) =
begin
  bind SetUUO = #075, SetDef = #033;
  local block[3];
  block[0] = 0^18 + 0;
  block[1] = .Protection and #777;
  block[2] = 0;  ! not mentioned in the monitor calls manual!
  vres = SetDef^18 + block<0,0>;
  return ifskirn calli(vres, SetUUO) then Yes else No;
end;

routine GetSwitchIniProtections =
begin
  local Prot;
  bind GetPPN = #024;  ! MC 4.5.10 p 4-10
  bind PJob = #030;  ! MC 4.5.22 p 4-14
  MyPPN = ifskirn calli(vres, GetPPN) then .vres else .vres;
  MyJob = (calli(vres, PJob)); .vres);
  ReadSwitches(sixbit 'LOGIN ');
  if FindAttr(sixbit 'DEFFPRO', No) then begin
    Self = ToOctal(.BindName(head(ThisAttrPtr)))
  end else begin
    Self = #055
  end % if login/defprot:xyz %;
  Group = .Self and #007;  ! force self&group to all rights
  Other = #000;  ! everyone has all rights
  Scan = if FindAttr(sixbit 'SCAN ', No) then Yes
       else if FindAttr(sixbit 'NOSCAN', No) then No else Yes;
  ReadSwitches(ProgramName);
  if FindAttr(sixbit "PROTEC", No) then begin
    if FindBind(sixbit "ASIS ") then begin
      Self = Group = Other = GetProtectionDefault();
    return
    end % as is %;
    SetNumber(sixbit "SELF ", Self, 8, .Self");
    SetNumber(sixbit "GROUP ", Group, 8, .Group");
  end;
SetNumber(sixbit "OTHER ", Other, 8, .Other);  ! either
SetNumber(sixbit "OTHERS", Other, 8, .Other);  ! spelling
end;
end;

routine NewProtectionDefault(NewPPN) =
  if .NewPPN< 0,18> eq1 .MyPPN< 0,18> then .Self else
  if .NewPPN< 18,18> eq1 .MyPPN< 18, 18> then .Group else
    % .NewPPN< 0,36> nea .MyPPN< 0,36> so % .Other;

=================================================================================

A Path Block (MC 8.1,10.9) is a ten-word block of the form

<table>
<thead>
<tr>
<th>Job number</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>switches and flags</td>
<td></td>
</tr>
<tr>
<td>Project nbr</td>
<td>Programmer #</td>
</tr>
<tr>
<td>sub-file-directory name 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>sub-file-directory name 6</td>
<td></td>
</tr>
<tr>
<td>000000</td>
<td>000000</td>
</tr>
</tbody>
</table>

CD maintains two path blocks: OldDefault (which is read), and NewDefault (which is written). There used to be two more blocks for tinkering with the library path, but they were removed as it was impossible to reconcile them with the complexity of /search logical names under TOPS10 version 7.01 (of which more below).

Each block has its opcode fixed, so DoPath only needs to be told which block to use. The old block is found by asking the monitor, and is copied into the new one. The new one is then modified in accordance with the command line and the user's SWITCH.INI file, and is then written back. This is CD's principal purpose. If the write-back fails (after strenuous attempts to find the new path), the original path is copied back into the new block and the initial path is restored. The report which tells the user where he is is always taken directly from the monitor in case of bugs.

The latest change to CD is to make it work in with PATH. In 7.01 you can define a logical name to stand for a sequence of paths, e.g. .PATH FRED:=DSKA:[]"JIM", FRED:DSK[][77:1234: FRED]. The intention is that CD FRED should take you to that place. To start with, we have to make an arbitrary choice: CD will try the first path in the list and no other, not even a modified form as with ordinary search. Secondly, CD ignores the device; so it won't always work even when
the path exists. CD **never** tinkers with the search list, which it would have to in this case. Finally, we have to use yet another form of path block and .PATH monitor call; see the new MC 27-227. We still use DoPath to read the block, but it had to be bent. CD will never write a new-style block, only read it (that's bad enough!)

```
macro
JobNo(PathBlock) = (PathBlock)[0]<18,18> $,
OpVal(PathBlock) = (PathBlock)[0]<0,18> $,
Flag(PathBlock) = (PathBlock)[1]$,
ScanSwitch = 0,2 $,
PPNof(PathBlock) = (PathBlock)[2]$,
ProjectNbr = 18,18 $,
Programmer = 0,18 $,
SfdOf(PathBlock,N) = (PathBlock)[2+(N)] $;
bind
MaxSdfs = 5, ! this MAY be different elsewhere
DoScan = 2, ! scan switch settings
DontScan = 1, ! scan switch settings
GetDefault = -1, ! Path opcode - read default path
SetDefault = -2, ! Path opcode - replace default path
GetLogical = -6, ! Path opcode - inspect logical name
own
OldDefault[10] = (GetDefault, 9:0), ! original default path
NewDefault[20] = (SetDefault, 29:0), ! modified default path
TryLogical[20] = (GetLogical, i^33, 18:0); ! logical name expansion
bind
LogticalBlk = TryLogical[5]; ! proper path part
own
LoseTpr, ! first Sfd which can't be squeezed out
NextSfd, ! place to put next sfd-name in NewDefault.
DirWanted; ! /dir switch given; simulate .Direct/Fast.
```

```
routine PrintPath(NewOld, PathBlock) =
begin
putext('"Your"
putix([NewOld]);
putexit('" default path is [";

putoct([PPNof([PathBlock]<ProjectNbr>));
putasc("*");
putoct([PPNof([PathBlock]<Programmer>));
incr i from 1 to 6 do begin
  if .SfdOf([PathBlock, i] eq 0 then exitloop;
  putasc("*");
  putix([.SfdOf([PathBlock, i]));
end;
putasc(""));

select .Flag([PathBlock]<ScanSwitch) of nset
  DoScan: putexit('"/scan");
  DontScan: putexit('"/noscan");
otherwise: putexit('"/??scan");
```
routine DoPath(PathBlock) =
  besin
  bind PathUUO = $110;
  ! JobNo(,PathBlock) always eal -1, which is invalid so => this one
  OPVal1(,PathBlock) always set to OPCode
  vres = (if ,%OPVal1(,PathBlock) eal GetLogical then 20 else 10)^16
       + ,PathBlock:<0,0>; ! [length,address]
  return ifskip calli(vres, PathUUO) then Yes else No
end;

routine ConsultMonitor(NewOld) =
  besin
  Protection = GetProtectionDefault() or 1^35;
  DoPath(OldDefault);

  if .NewOld nea sixbit 'NEW' then return;

  putext('[ Now at ]';
  putoct(.PPNo(OldDefault)<ProjectNbr>);
  putasc('"');
  putoct(.PPNo(OldDefault)<Programmer>);
  incr i from 1 to MaxSfdes do besin
    if ,SfdOf(OldDefault, ,i) eal 0 then exitloop;
    putasc(';');
    putsix(.SfdOf(OldDefault, ,i));
  end;
  putext('[ ] <'');
  putasc('0'*+,Protection<6,3>);
  putasc('0'*+,Protection<3,3>);
  putasc('0'*+,Protection<0,3>);
  putasc('>');
  if .Flag(OldDefault)<ScanSwitch> nea Doscan then putext('/noscan');
  if .TryLogical[4] nea 0 then putext('(logical name)');
  putext(' ]?M?J');
end;

=================================================================

The <base> of a path may take one of these forms:

<base> ::= <identifier>  {a}
  | -  {b}
  | ~  {c}
  | +  {d}
  | <project> {, <programmer>}  {e}

where

<project> ::= <octal number> | <empty>
<programmer> ::= <octal number> | <empty>

To take the simplest case first, {e} a project-programmer number
pair stands for itself. An empty project or programmer number is
filled in from the corresponding field of your LOGIN PPN. If the
programmer number is empty, and there are no SFDs following, both
it and its comma may be omitted. Thus if the current job has PPN
[123,456], then [123,456], [,456], [123,], [123], [,], and [] all
mean the same, while [777] means [777,456], and [,702] [123,702].

The next simplest case is {d}. The plus sign means to use just
the PPN of the path. Thus if you are currently in the directory
[1234,5670,jim,fred] then the <base> [+ means "[1234,5670]."

Case {b} specifies all of the current path. If you don’t want CD
to search, then e.g. to continue the example above, [{-,harry stands}
for the path [1234,5670]+jim,fred,harry]. "CD -" is a good way of
checking what your current path is (like UNIX’s "pwd"), and CD -/dir
is an alternative to DIR/FAST.

Case {c} specifies the current path with the last sfd dropped, if
there is one. The up arrow is meant to be suggestive of motion UP
the directory tree. You may have SFD snames after this too, like all
bases, but there isn’t much point. CD ^FRED will take you to just
the same place as CD FRED unless there is a FRED in the current path.

Cases {b.,e} are explicit paths; if CD can’t find the path you’ve
described it will give up at once.

The interpretation of case {a} depends on your SWITCH.INI file. If
it has a line CD/NAME:<<identifier>>=<path> the identifier means
whatever <path> means. Thus if "CD/NAME:RUFUS=[,,SWORDS,GRAM],"
then RUFUS,HILT means [,,SWORDS,GRAM,HILT]. On the other hand, if
the <identifier> is not defined in your SWITCH.INI file, CD will go
and look for it, starting in the current SFD, working up the tree,
and then finally dropping back into your UFD. Thus if you are in
the directory [1234,5670,jim,fred] and give the command ,CD harry
CD will try [1234,5670,jim,fred,harry], [1234,5670,jim,harry],
[1234,5670,harry], then [{-,harry} in that order, stopping when it
finds it. If harry still can’t be found, CD will check to see if
HARRY is defined as a logical path name, and so there if it is.

UNIX equivalences:

<table>
<thead>
<tr>
<th>UNIX</th>
<th>CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd a/b/c</td>
<td>cd a,b,c</td>
</tr>
<tr>
<td>cd</td>
<td>cd</td>
</tr>
<tr>
<td>cd .</td>
<td>cd</td>
</tr>
<tr>
<td>cd ..</td>
<td>cd</td>
</tr>
<tr>
<td>cd *name</td>
<td>cd name</td>
</tr>
<tr>
<td>cd ***; ls</td>
<td>cd ***/dir</td>
</tr>
</tbody>
</table>

======================================================================

forward ParsePath;

routine Base =
  begin
    ch = setstr(TextPtr, SixBuf);
    if \ch eq "[" then begin
      if .SixBuf neq 0 then PutError(1, 7);
      ch = setstr(TextPtr, SixBuf);
      end % if "[" %;

    LoseTop = SfdOf(NewDefault, 0);
    NextSfd = SfdOf(NewDefault, 1);
if .ch eal "+" or .ch eal "-" or .ch eal "^" then besin
decr i from MaxSfds+1 to 0 do
  SfdOf(NewDefault, .i) = SfdOf(OldDefault, .i);
while .".NextSfd neq 0 do NextSfd = .NextSfd+1;
if .ch eal "+" then besin
  NextSfd = SfdOf(NewDefault, 1)
end else
if .ch eal "^" then besin
  if .NextSfd neq SfdOf(NewDefault, 1) then NextSfd = .NextSfd-1
end % if + or - %
if .SixBuf neq 0 then PutError(1, 8);
  ch = setsix(TextPtr, SixBuf);
if .SixBuf neq 0 then PutError(1, 18);
end % + - % else
if .SixBuf<30,6> see sixbit "A" and .SixBuf<30,6> see sixbit "Z" then besin
  if FindBoth(sixbit "NAME ", .SixBuf) then besin
    local SaveCh, SaveTextPtr;
    bind BadDef = uplit asciz '!!undefined!!';
    SaveCh = .ch;
    SaveTextPtr = .TextPtr;
    Strings(.SixBuf, TextPtr, BadDef<36,7>);
    ParsePath();
    ch = .SaveCh;
    TextPtr = .SaveTextPtr;
  end
end else besin
  decr i from MaxSfds+1 to 0 do
    Sfdof(NewDefault, .i) = SfdOf(OldDefault, .i);
  while .".NextSfd neq 0 do NextSfd = .NextSfd+1;
  (LoseToP = .NextSfd) = .SixBuf;
  NextSfd = .NextSfd+1;
end
end % name % else
if .ch eal "," or .ch eal "]" or .ch eal EOS then besin
local pn;
PPNof(NewDefault) = .MyPPN;
pn = ToOctal(.SixBuf);
if .pn neq 0 then PPNof(NewDefault)<ProjectNbr> = .pn;
if .ch eal "," then besin
  ch = setsix(TextPtr, SixBuf);
  pn = ToOctal(.SixBuf);
  if .pn neq 0 then PPNof(NewDefault)<Programmer> = .pn;
end % "," %
end % cases {a,b,c,d,e} %
if .ch neq "," and .ch neq "]" and .ch neq "<" and
  .ch neq "/" and .ch neq EOS then PutError(2, 9);
end;

routine ParsePath =
besin
  Base();
  while .ch eal "," do besin
    ch = setsix(TextPtr, SixBuf);
  if .SixBuf eal 0 then besin
    PutError(2, 10);
  end else besin
if .NextSfd-.LoseTop str MaxSfds then begin
    PutError(1, 16);
    NextSfd = .LoseTop-1;
end % overflow check %;
.NextSfd = .SixBuf;
NextSfd = .NextSfd+1;
end;
end % while %;
if .ch eal "J" then begin
    ch = setsix(TextPtr, SixBuf);
    if .SixBuf neq 0 then PutError(1, 11);
end;
.NextSfd = 0;
if .ch eal "<" then begin
    ch = setint(TextPtr, 8, Protection);
    if .ch eal ">" then begin
        ch = setsix(TextPtr, SixBuf);
        if .SixBuf neq 0 then PutError(1, 12);
    end else begin
        PutError(2, 13);
    end;
end % protection %;
if .ch eal "/" then begin
    FindAttr(sixbit "SWITCH", Yes);
    do value(Yes) until .ch neq "/";
end;
if .ch neq EOS then PutError(2, 14);
end;
!
CheckPath makes sure that there aren't too many Sfds in the
NewDefault path.  If there are, something like CD A,B,C,D,E
must have happened in a deep directory, so it is enough to
squeeze out what we can, as if long things weren't "found".

routine CheckPath =
begin
    local Size, Lose, Shft;
    Size = .NextSfd-SfdOf(NewDefault, 1);
    Lose = .LoseTop-SfdOf(NewDefault, 0);
    if .Size leq MaxSfds then return Yes;
    if .Size-lose str MaxSfds then begin
        SixBuf = .SfdOf(NewDefault, .Size);
        PutError(1, 16);
        return No;
    end % if %;
    Shft = .Size-MaxSfds;
    incr i from .LoseTop to .NextSfd do (.i-,.Shft) = ..i;
    LoseTop = .LoseTop-.Shft;
    NextSfd = .NextSfd-.Shft;
    return Yes;
end % CheckPath %;

routine SetPathFlag =
begin
    bind LSCN = .scan;      ! quick hack
    local scan, help;
    scan = LSCN; ! default to the login/scan settings
    help = No; ! don't be overly helpful unless asked
    DirWanted=No; ! don't display the new directory unless asked
    if FindAttr( sixbit "SWITCH", No) then begin
SetSwitch(sixbit "SCAN ", scan);
SetSwitch(sixbit "HELP ", help);
SetSwitch(sixbit "DIR ", DirWanted);
end;
if .scan eq1 Yes then Flasg(NewDefault)<ScanSwitch> = DoScan else
if .scan eq1 No then Flasg(NewDefault)<ScanSwitch> = DontScan;
if .help neq No then GiveHelp(ProgramName, .help, 0);
end;

routine ReadCommand =
begin
  decr i from 9 to 1 do begin
    NewDefault[,] = .OldDefault[,] ;
  end;
  TextPtr = GetTermLine(Yes, LineBuf);
  ParsePath();
  CheckPath();
  SetPathFlag();
  %< PrintPath(sixbit 'TMP', NewDefault);
  %>
end;

routine Check701LogicalName =
begin
  if .TryLogical[1] eq1 0 then return No;  ! already tried
  TryLogical[1] = 133;  ! flash cleared by call, alas!
  TryLogical[2] = .SfdOf(NewDefault, 1);  ! The name
  TryLogical[3] = 0;
  if not DoPath(TryLogical) then return No;
  incr i from 1 to 6 do
    if .SfdOf(LogicalBlk, ,i) eq1 0 then begin
      % check for single definition %
      if .SfdOf(LogicalBlk, ,i+2) neq 0 then PutError(3, 19);
      % append rest of NewDefault %
      incr j from 2 to 6 do
        if (SfdOf(LogicalBlk, ,i+ ,j-2) = .SfdOf(NewDefault, ,j)) eq1 0
          then exitloop % j %
      exitloop % i %
    end % if %;
    % now move the new path to NewDefault and check it %
    incr i from 0 to 6 do SfdOf(NewDefault, ,i) = .SfdOf(LogicalBlk, ,i);
    CheckPath();
    return .ErrorCount eq1 0
  end;

routine ObeyCommand =
begin
  TryLogical[4] = 0;
  if .ErrorCount eq1 0 then begin
    until DoPath(NewDefault) do begin
      if .LoseTop eq1 SfdOf(NewDefault, 0) then begin
        if not Check701LogicalName() then begin
          % nothing else left to try, so back %
          PutError(0, 22);  ! don’t forget not to change protection
          decr i from 9 to 1 do NewDefault[,] = .OldDefault[,] ;
        end % if %
      end else
      if .LoseTop eq1 SfdOf(NewDefault, 1) then begin
        % try looking in Job’s PPN %
    end;
end;
LoseTop = SfdOf(NewDefault, 0);
PPNof(NewDefault) = .MyPPN;
end else begin
    incr Sfd from .LoseTop-1 to .NextSfd do
        .Sfd = (.Sfd+1); ! move everythings down one
    NextSfd = .NextSfd-1;
    LoseTop = .LoseTop-1;
end % if %;
end % until %;
end % if %;
if .ErrorCount % now % neg 0 then begin
    decr i from 9 to 1 do begin
        NewDefault[,i] = .OldDefault[,i];
    end;
    DoPath(NewDefault);
end else begin
    if .Protection % still % lss 0 then
        Protection = NewProtectionDefault(.PPNof(NewDefault));
    if not SetProtectionDefault(.Protection) then PutError(0, 23);
end;
end;
bind Reset = $00, ExitI = $12; ! calli codes
GetSwitchIniProtections();

while 1 do begin
    calli(0, Reset);
    ErrorCount = 0;
    ConsultMonitor(sixbit 'Old');
    ReadCommand();
    ObeyCommand();
    ConsultMonitor(sixbit 'New');
    if .DirWanted then FastDirectory(OldDefault);
    calli(1, ExitI); ! don't type the message
end % forever %;
end
'udom
This program reads a Prosos source file and extracts the tokens. The tokens are written on the output or counted and sorted.

The fact which make the character "(*) a funny one is that if it is
Preceded by a <word>, <quoted>, or <operator> it represents the
token ' ('; otherwise it represents the token '('.
The fact which makes the character '.' funny too is that if it is
followed by a <layout> character it represents the token '.', otherwise it represents the token '.', .

*******************************************************************************
Program wlist;

label
9;
{end of program}

const
MaxTok = 1280;
{longest token that can be handled}
MaxHash = 2003;
{size of dictionary (Prime)}

type
HtIndex = 1..MaxHash;

StIndex = 1..MaxTok;
{index into a string text buffer}
Slength = 0..MaxTok;
foobaz = packed array [StIndex] of char;
Strings = record
tally: integer;
{how often seen}
length: Slength;
{number of characters}
offset: 0..MaxTok;
{offset from start of buffer}
buffer: ^foobaz;
end {strings};

StringsP = ^Strings;

toclass = (int {integer constant}
,str {string constant}
,vbl {variable}
,wd {ordinary word}
,op {operator: made of signs}
,otd {quoted atom}
,pct {punctuation}
,frag) {display frequencies?}

chclass = (digit, lower, upper, single, sign,
,comment1, comment2, quoted1, quoted2,
,funny1, funny2, spacing, newline);

var
dictionary: array [HtIndex] of StringsP;
nwords: 0..MaxHash;
heaptor: integer;
infile, outfile, helDfile: text;
MinTally, MaxTally, MinCount: integer;
upcase: array [char] of char;
notice: array [toclass] of boolean;
ctype: array [char] of chclass;
nulch, eofch: char;
chhash: array [char] of integer;
curbuf: Strings;

function CommandLine: integer;
1190     label
1191         9;
1192
1193     const
1194         MaxSpec = 72;         \{longest possible file specification\}
1195
1196     type
1197         chcnt = 0..MaxSpec;
1198         chnum = 1..MaxSpec;
1199         filespec = record length: chcnt; text: packed array [chnum] of char end;
1200         charset = set of char;
1201
1202     var
1203         result: 0..2;
1204         inspec, outspec: filespec;
1205         ch: char;
1206
1207     procedure readspec(var spec: filespec; delimiters: charset);
1208        label
1209             9;
1210     var
1211         i: chcnt;
1212
1213     begin
1214         i := 0;
1215         while not eoln(tty) do begin
1216             read(tty, ch);
1217             if ch in delimiters then goto 9;
1218             if upcase[ch] <> '"' then begin
1219                 i := i+1;
1220                 spec.text[i] := upcase[ch];
1221             end {if}
1222         end {while};
1223         ch := chr(10); \{"J = line feed\}
1224         9;
1225         spec.length := i;
1226         while i < MaxSpec do begin
1227             i := i+1;
1228             spec.text[i] := '"';
1229         end {while};
1230     end;
1231
1232     procedure GiveHelp;
1233     var
1234         c: char;
1235
1236     begin
1237         reset(helpfile, 'MEX\WLIST.HLP');
1238         while not eof(helpfile) do begin
1239             read(helpfile, ch);
1240             write(tty, ch);
1241         end {copying MEX\WList.HLP to Tty:};
1242         close(helpfile);
1243         result := 1;
1244     end;
1245
1246     procedure readswitch;
1247     var
1248         switch: filespec;         \{the switch text following /\}
1249         b: boolean;          \{value for logical switches\}
1250         n: integer;         \{value for integer switches\}
1251         c: char;
1252         j: chnum;
begi
readspec(switch, ['/']);
for j := 1 to switch.length do begin
  c := switch.text[j];
  if (c >='0') and (c <= '9') then
    n := n*10 + (ord(c)-ord('0'));
end {for};

b := true; {default /foobaz = /foobaz:yes}
c := switch.text[1];
if c = 'N' then begin
  c := switch.text[3];
b := false; {/NOfoobaz = /foobaz: no}
end;

end {case}

begi
result := 0;
MinTally := 1;
MaxTally := 1000000000;
MinCount := 1;
notice[int] := false; notice[str] := false;
notice[vbl] := true; notice[wrd] := true;
notice[op ] := false; notice[atd] := true;
notice[pct] := false; notice[fra] := true;
write(tty, ' *');
readln(tty); {seems to be needed}
if eof(tty) then begin
  result := 2;
soto 9
end; {if "Z typed};
readspec(outspec, ['=', '_']);
if ch = chr(10) then begin
  writeln(tty, '" expected after output file spec ",
    outspec.text[outspec.length, '"');
  result := 1;
  goto 9;
end; {if};
readspec(inspec, ["/"]);
if inspec.length = 0 then begin
  writeln(tty, 'No input file specified');
  result := 1;
  goto 9;
end;
if outspec.length = 0 then begin
  outspec.length := 4;
end;
while ch = '/' do readswitch;
reset(infile, inspec.text);
rewrite(outfile, outspec.text);
)
9: CommandLine := result;

procedure ReadlnPut;
label
1;
{used for setting to 'sign' from '/'}

var
ch, heldch: char; {current, lookahead characters}
quote: char; {" / or eofch if not in a string}
functorseen: boolean; {to disambiguate " -> ( / \ )}
token: array [StIndex] of char; {current token, not packed}
toklen: integer; {allow for possible overflow}
k: HtIndex; {Just for clearing the dictionary}

procedure LookUP;
label
1; {collision}
9; {found}

var
j: StIndex;
k: HtIndex;
h: integer;
begin
h := 0;
for j := 1 to toklen do
  h := (h*59 + chhash[token[j]]) mod MaxHash;
h := h+1;
k := h;
repeat
  if dictionary[k] = NIL then begin
    if curbuf.length < toklen then with curbuf do begin
      new(buffer); {allocate another page}
      length := MaxTok;
      offset := 0;
      end {checking for buffer overflow};
new(dictionary[k]);
with dictionary[k] do begin
  tally := 1;  // seen this token once
  length := toklen;
  offset := curbuf.offset;
  buffer := curbuf.buffer;
end {dictionary entry};

with curbuf do begin
  for j := 1 to toklen do buffer[offset+j] := token[j];
  length := length-toklen;
  offset := offset+toklen;
end {copying string};
soto 9;
end {entered};

with dictionary[k] do begin
  if length <> toklen then soto 1;
  for j := 1 to toklen do
    if buffer[offset+j] <> token[j] then soto 1;
  tally := tally+1;
soto 9;
end {found};

if k = MaxHash then k := 1 else k := k+1;
until k = h;
writeln(tty, 'Dictionary overflow with', MaxHash, ' words');
k := k-h;           // bomb out
end {Look up};

procedure nextch;
begin
  if heldch <> nulch then begin
    ch := heldch;
    heldch := nulch;
  end else
    if eof(infile) then begin
      ch := chr(26);
    end else begin
      read(infile, ch);
    end
end {next character};

procedure savech;
begin
  heldch := ch;
end;

procedure nextat;
begin
  nextch;
  if ch = eofch then begin
    savech;
    ch := nulch;
  end else
    if ch = quote then begin
      nextch;
      if ch <> quote then begin
        savech;
        ch := nulch;
      end
    end
end
end {next quoted character};

Procedure stash:
  var J: 1..60; {on overflow, print just a prefix of the token}
  begin
    toklen := toklen+1;
    if toklen <= MaxTok then begin
      token[toklen] := ch
    end else
    if toklen = MaxTok+1 then begin
      write(tty, '*** token too long - ');  
      for J := 1 to 60 do write(tty, token[J]);
      writeln(tty, '...');
    end {overflow};
    nextat;
  end {stash};

Procedure finished(sort: toclass);
  var J: StIndex;
  begin
    if notice[sort] then begin
      if notice[frac] then begin
        if toklen > MaxTok then toklen := MaxTok;
        LookUp;
      end end
      for J := 1 to toklen do write(outfile, token[J]);
      writeln(outfile)
    end;
    functorseen := (sort = wrd) or (sort = op) or (sort = atd);
    toklen := 0;
  end;

begin
  for k := 1 to MaxHash do dictionary[k] := NIL;
  with curbuf do begin
    new(buffer); {allocate a new page of characters}
    length := MaxTok; {all of it is free}
    offset := 0; {none of it has been used}
  end {with};
  toklen := 0; {haven't started a word yet}
  quote := eofch; {not in a string or quoted atom}
  ch := ' '; {will cause functorseen to be cleared}
  while ch <> eofch do
    case chtype[ch] of
      digit:
        begin
          repeat stash until chtype[ch] <> digit;
          if ch = '/' then
            repeat stash until chtype[ch] <> digit;
            finished(int);
        end {integer};
      lower:
        begin
          repeat stash until chtype[ch] > upper;
          finished(wrd);
        end {word};
```plaintext
upper: begin
    repeat stash until chtype[ch] > upper;
    finished(vbl);
end {variable};

quote1: begin
    quote := ch; nextat;
    while ch <> nullch do stash;
    quote := eofch; nextat;
    finished(alt);
end {quoted};

quote2: begin
    quote := ch; nextat;
    while ch <> nullch do stash;
    quote := eofch; nextat;
    finished(str);
end {string};

comment1: begin
    repeat nextch until chtype[ch] = newline;
    functorseen := false;
end {percent comment};

comment2: begin
    quote := ch; nextat;
    if ch <> '*' then begin
        savech; ch := quote;
        soto 1
    end {if not a comment after all};
    repeat
        if ch = '*' then begin
            nextch;
            if ch = quote then ch := nullch;
        end else begin
            nextch
            end {looking at *X or Y} until (ch = nullch) or (ch = eofch);
    end {lookin!:I at *X or Y}
    if ch = nullch then ch := ' ';
    quote := eofch;
    functorseen := false;
end {/ * ... */ comment};

sign: begin
    repeat stash until chtype[ch] <> sign;
    finished(op);
end {operator};

single: begin
    stash;
    finished(pct);
end {punctuation};

funny: begin
    if functorseen then begin
        savech; ch := ' '; stash;
    end {if functor application};
    stash;
    finished(pct);
```

funny2:
begin
  stash;
  if chtype[ch] >= spacins then begin
    ch := ' ';
    stash;
  end {terminal period};
  finished(pct);
end {period};

spacins,
newline:
begin
  nextch;
  functorseen := false;
end {case and while};

end {Read Input};

Procedure Pack;
var
  k: HtIndex;
begin
  nwords := 0;
  for k := 1 to MaxHash do
    if dictionary[k] <> NIL then
      if (dictionary[k].tally >= MinTally) and
          (dictionary[k].tally <= MaxTally) then begin
        nwords := nwords+1;
        dictionary[nwords] := dictionary[k];
      end {if keeping entry};
end {Packing the dictionary};

Procedure HeapSort;
var
  k: HtIndex;
  entry: StringsP;
begin
  function greater(var b1, b2: foobaz; o1, o2: Slensth;
                   const) 11, 12: Slensth): boolean;
    label 9;
    var L: Slensth;
    begin
      L := 11;
      if L > 12 then L := 12;
      while L > 0 do begin
        o1 := o1+1; o2 := o2+1;
        if b1[o1] <> b2[o2] then begin
          greater := b1[o1] > b2[o2];
          soto 9;
        end {different};
        L := L-1;
      end {for L := L downto 1};
      soto 9;
    end {greater 1};
function greater(w1, w2: StringsP): boolean;
begin
greater := greater1(w1^,buffer^, w2^,buffer^,
    w1^,offset, w2^,offset, w1^,length, w2^,length);
end {w1 > w2?};

Procedure RestoreHeap(l, n: HtIndex);
  label 9;
  var k, m: HtIndex;
  begin
    entry := dictionary[l];
    k := l;
    while k*2 <= n do begin
      m := 2*k;
      if m < n then
        if greater(dictionary[m+1], dictionary[m]) then
          m := m+1;
        end {if greater};
      if not greater(dictionary[m], entry) then goto 9;
      dictionary[k] := dictionary[m];
      k := m;
      end {while};
    9:
    dictionary[k] := entry;
  end {restoring a heap};

begin { sorting the packed dictionary }
  { make the dictionary into a heap }
  for k := nwords div 2 downto 1 do RestoreHeap(k, nwords);
  { sort the heap }
  for k := nwords downto 2 do begin
    entry := dictionary[l];
    dictionary[l] := dictionary[k];
    dictionary[k] := entry;
    RestoreHeap(l, k-1);
  end {for k};
end {Heap Sort};

Procedure PrintOut;
  var
    k: HtIndex;
    J: StIndex;
    t: integer;
    a: boolean; {for writea effect}
  begin
    t := 0;
    for k := 1 to nwords do with dictionary[k] do begin
      t := t+tally;
      if tally > MinCount then begin
        write(outfile, tally:4, ' ')
      end else begin
        write(outfile, ' ')
      end {if showing count};
      a := false;
      if length <= 1 then begin
        end else if chtype[buffer^[offset+1]] <= upper then begin
          for J := 2 to length do
            if chtype[buffer^[offset+J]] > upper then a := true;
        end else begin
          for J := 2 to length do
            if chtype[buffer^[offset+J]] <> sign then a := true;
          end {if chtype[buffer^[offset+1]] <= upper};
        end else begin
          for J := 2 to length do
            if chtype[buffer^[offset+J]] > upper then a := true;
        end {if chtype[buffer^[offset+1]] <= upper};
      end else begin
        for J := 2 to length do
          if chtype[buffer^[offset+J]] <> sign then a := true;
        end {if chtype[buffer^[offset+1]] <= upper};
      end else if chtype[buffer^[offset+1]] <= upper then begin
        for J := 2 to length do
          if chtype[buffer^[offset+J]] > upper then a := true;
      end else begin
        for J := 2 to length do
          if chtype[buffer^[offset+J]] <> sign then a := true;
      end {if chtype[buffer^[offset+1]] <= upper};
    end {for k};
end {PrintOut};
5990     end;
6000     if a then write(outfile, '"''
6010     for j := 1 to length do write(outfile, buffer[offset+j]);
6020     if a then write(outfile, '"''
6030     writeln(outfile);
6040     end {for k};
6050     break(outfile); {in case it is tty}
6060     writeln(tty, nwords:4, ',', t:1, ' types\tokens');
6070     end {Print Out};

6080
6090
6100     procedure Initialise;
6110     var ch: char; k: integer;
6120     begin
6130     for ch := chr(0) to chr(127) do upcase[ch] := ' ';
6140     for ch := 'A' to 'Z' do upcase[ch] := chr(ord(ch)-32);
6150     for ch := '0' to '9' do upcase[ch] := ch;
6160     upcase[':'] := ':';
6170     upcase['\'] := '\';
6180     upcase['\'] := '\';
6190     upcase['\'] := '\';
6200     nulch := chr(0);
6210     eofch := chr(26);
6220     for ch := chr(0) to chr(127) do chtype[ch] := sign;
6230     for ch := '0' to '9' do chtype[ch] := digit;
6240     for ch := 'A' to 'Z' do chtype[ch] := lower;
6250     for ch := 'A' to 'Z' do chtype[ch] := upper;
6260     chtype['-'] := upper;
6270     chtype['.' ] := comment1;
6280     chtype['/'] := comment2;
6290     chtype[''''] := quote1;
6300     chtype['\'] := quote2;
6310     chtype['('] := funny1;
6320     chtype[')'] := funny2;
6330     chtype['{'] := single; chtype['}'] := single;
6340     chtype['['] := single; chtype[']'] := single;
6350     chtype[']'] := single; chtype[']'] := single;
6360     chtype[']'] := single; chtype[']'] := single;
6370     chtype[chr(7)] := newline; chtype[chr(9)] := spacing;
6380     chtype[chr(10)] := newline; chtype[chr(11)] := newline;
6390     chtype[chr(12)] := newline; chtype[chr(13)] := spacing;
6400     chtype[chr(26)] := newline; chtype[chr(27)] := newline;
6410     chtype[chr(31)] := newline; chtype[chr(32)] := spacing;
6420     for ch := chr(0) to chr(127) do chhash[ch] := 0;
6430     for ch := '0' to '9' do chhash[ch] := ord(ch)-ord('0')+1;
6440     for ch := 'a' to 'z' do chhash[ch] := ord(ch)-ord('a')+11;
6450     for ch := 'A' to 'Z' do chhash[ch] := ord(ch)-ord('A')+37;
6460     chhash['-'] := 63;
6470     k := 63;
6480     for ch := chr(127) downto chr(0) do if chhash[ch] = 0 then begin
6490     end {initialisation};
6500     begin {program}
6510     Initialise;
6520     while true do begin
6530     case CommandLine of
begin
  mark(heapTop);
  ReadInput;
  if notice[frag] then begin
    Pack;
    if nwords = 0 then begin
      writeln(tty, '** no tokens');
      end else begin
      Heapsort;
      PrintOut;
      end {if needed by USELESS Pascal};
    end {sort and print};
    close(outfile);
    close(infile);
    release(heapTop);
  end {normal case};
1: begin
  end {error in command line};
2: goto 9;
end {case};
end {forever};
end {program};

No error detected

Highs:  5K
Lows:   3K