58

AIX update

August 2000

In this issue

AIX Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 550955
From USA: 01144 1635 33823
E-mail: harryl@xephon.com

North American office

Xephon/QNA
Post Office Box 350100, Westminster CO 80035-0100, USA
Telephone: (303) 410 9344
Fax: (303) 438 0290

Contributions

If you have anything original to say about AIX, or any interesting experience to recount, why not spend an hour or two putting it on paper? The article need not be very long – two or three paragraphs could be sufficient. Not only will you actively be helping the free exchange of information, which benefits all AIX users, but you will also gain professional recognition for your expertise and that of your colleagues, as well as being paid a publication fee – Xephon pays at the rate of £170 ($250) per 1000 words for original material published in AIX Update.

To find out more about contributing an article, see *Notes for contributors* on Xephon's Web site, where you can download *Notes for contributors* in either text form or as an Adobe Acrobat file.

Editor

Harold Lewis

Disclaimer

Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, scripts, and other contents of this journal before making any use of it.

Subscriptions and back-issues

A year's subscription to *AIX Update*, comprising twelve monthly issues, costs £180.00 in the UK; $275.00 in the USA and Canada; £186.00 in Europe; £192.00 in Australasia and Japan; and £190.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the November 1995 issue, are available separately to subscribers for £16.00 ($23.00) each including postage.

AIX Update on-line

Code from *AIX Update* is available from Xephon's Web page at *www.xephon.com/aixupdate.html* (you'll need the user-id shown on your address label to access it).

An administrator’s introduction to make

The **make** utility is frequently thought of as a programmer’s tool (if it is thought of at all), but it has applications outside the world of programming.

In this article, I’ll first cover exactly what **make** is and then show you some real-world uses of it. **make** allows you to create, recreate, and update a file based on the existence and/or modification time of one or more other files. The simplest way to understand **make** is to look at a programming example.

In C programming, a source file may be created with **vi** or another suitable text editor. This file usually has the extension *.c*, so we’ll start with an example called *hello.c*. The source file is compiled into an object file that usually has the extension *.o*, so in this case the object file would be *hello.o*. The object file is then linked into a running program called **hello** (with no extension).

In terms of the **make** utility, the target *hello.o* depends on *hello.c*, and the target **hello** depends on *hello.o*.

A **make** file is a script-like file that describes this target-dependency relationship, also providing the commands needed to create or recreate one target file from one or more other dependent files. The target, the dependence relationship, and the command constitute a **make** *rule*. **make** rules for this simple example are shown below.

```
hello.o : hello.c
   $(CC) -c hello.c

hello : hello.o
   $(CC) hello.o -o hello
```

In the first rule, *hello.o* appears on a line followed by a colon (‘:’) and *hello.c*. Note the spaces around the colon, which are required by some versions of **make**. This is the **make** syntax used to indicate that *hello.o* depends on *hello.c*. Underneath that line is the command:

```
$(CC) -c hello.c
```

which is executed to create *hello.o*. The variable *$(CC)* is a default variable in **make** that holds the name of the C compiler. This is defined when **make** is installed and is very handy considering that the C compiler may be called **cc**, **c89**, **gcc**, etc. The second rule is similar to the first, stating that **hello** depends on *hello.o* and that the command:

```
cc hello.o -o hello
```

is used to create **hello** from *hello.o*.

Without wishing to go deeper into C programming, I have included a simple version of *hello.c* below. If you have access to a C compiler, you can use it to try the examples in this article.

```
#include <stdio.h>

main()
{
    printf("Hello world.\n");
}
```

The **make** utility uses a default **make** file named, appropriately enough, *makefile* or *Makefile*. If you store the above listing in a file called *makefile* or *Makefile* and put it in a directory containing the source code *hello.c*, you could type the command:

```
make hello
```

and the **hello** program would be built. If the compile and link was successful, you may run your simple program by typing **./hello** on the command line.

```
./hello
Hello world.
$
```

If you type the command **make hello** a second time, a message is displayed indicating that the **hello** program is up-to-date:

```
make hello
make: 'hello' is up to date
$
```

But what does ‘up-to-date’ mean in this context? It means that **hello**’s modification date and time stamp is greater than or equal to the modification date and time stamp of all the files on which **hello** depends – in this case, *hello.o* and, ultimately, *hello.c*.

Whenever you run **make**, the program creates an internal table of dependencies, then verifies the creation or modification date and time stamps, and finally works out which files have to be created or recreated. This includes checking whether files exist. For example, the first time you run **make hello**, *hello.o* doesn't exist and is, therefore, considered out-of-date with respect to *hello.c*. After *hello.o is* created, **make** checks for **hello** and finds that it's also missing and is, therefore, considered out-of-date with respect to *hello.o*.

If a programmer modifies *hello.c*, its new modification date and time stamp cause *hello.o* to be out-of-date with respect to it.

If you've been using *hello.c* to follow this example, then open the file and **make** some small change to it, such as adding a comma after the word 'hello'. Save and close the file. You have now changed the file's modification date.

```
#include <stdio.h>

main()
{
    printf("Hello, world.\n");
}
```

If **make hello** were run again after this change, *hello.o* would be rebuilt from *hello.c*. Once *hello.o* is recreated, its modification date is changed, and so **hello** would be out of date with respect to *hello.o*, which causes **hello** to be rebuilt.

In this way **make** does two jobs: it checks whether a program or file needs to be rebuilt based on changes made to the original starting document or a dependency file, and it simplifies the commands that need to be executed to recreate the program or target file. **make hello** is a lot easier to remember and faster to type than **cc -c hello.c** followed by **cc hello.o -o hello**. In fact, by using the *$(CC)* variable, you don't even have to know the name of your C compiler, which might be something other than **cc**.

We'll now look at a more complex example that's outside the programming arena.

Assume that you've created a book containing five chapters named *chap1.txt* through *chap5.txt*. The process of assembling the chapters into a book involves:

1. Combining the chapters by concatenating them end-to-end.
2. Building a table of contents to place at the beginning of the book.
3. Paginating the resulting table of contents and chapters.
4. Building an index using the page numbers created in the previous step.
5. Appending the index to the end of the book.
6. Paginating again to number the index pages.

Let's also assume that you have four programs available: **cat** concatenates text files, **toc** builds a table of contents, **idx** builds an index, and **paginate** renumbers the pages.

The first version of this process uses several intermediate files. The rules for this process are shown in the following listing. There is a problem with this makefile that I will cover in a moment, but, as it stands, it illustrates several new things about **make** that you need to know. The makefile illustrated includes line numbers for explanation.

```
 1 # builds thebook.txt from the chapters
 2
 3 .SUFFIXES: .txt
 4
 5 allchaps.txt : chap1.txt chap2.txt chap3.txt chap4.txt chap5.txt
 6   cat chap1.txt chap2.txt chap3.txt chap4.txt chpa5.txt >allchaps.txt
 7
 8 tocchaps.txt : allchaps.txt
 9   toc allchaps.txt >toc.txt
10   cat toc.txt allchaps.txt >tocchaps.txt
11   paginate tocchaps.txt
12   rm -f toc.txt
13   rm -f allchaps.txt
14
15 thebook.txt : tocchaps.txt
16   idx tocchaps.txt >thebook.txt
17   paginate thebook.txt
18   rm -f tocchaps.txt
```

The first entry (line 3) is not a rule but a signal that *.txt* is to be added to the list of standard suffixes that **make** recognizes. The list of standard suffixes usually includes some standard file extensions, such as *.c* for C files, *.o* for object files, *.sh* for shell scripts, and so on. The *.txt* extension is not in the standard list and so needs to be added.

The rule after the *.SUFFIXES* entry (line 5) illustrates the fact that one target file may have more than one dependency. It states that the file *allchaps.txt* depends on the five chapter files. The command to create or update *allchaps.txt* (on line 6) uses **cat** to string the five chapters together into the one file, *allchaps.txt*.

The second rule (line 8) illustrates the fact that more than one command may be executed to create the required target file. It states that *tocchaps.txt* (table of contents plus chapters) depends on the existence of *allchaps.txt* and is created by running the **toc** program to create a table of contents (line 9) and then concatenating *toc.txt* and *allchaps.txt* to create *tocchaps.txt* (line 10). The result is paginated (line 11) and then the two temporary files, *toc.txt* and *allchaps.txt*, are removed (lines 12 and 13). This leaves a paginated file containing a table of contents and the chapters. The final rule (line 15) adds the index (line 16), repaginates the file (line 17), and leaves the output named *thebook.txt* (line 18).

If you have all the chapter files and this makefile in one directory and you type **make thebook.txt**, **make** will build *thebook.txt* for you. If you edit one of the chapters later on, the same command will rebuild your book with the new information.

Earlier I mentioned a problem with this makefile. The problem is that, even if you don't edit any of the chapters before issuing the command **make thebook.txt** once more, the whole process is executed again. Why? The simple answer is that the intermediate files are deleted. When you type the command, **make** goes to the rule for building *thebook.txt* and sees that it needs something called *tocchaps.txt*. However, *tocchaps.txt* was deleted on line 18. So **make** looks for the rule on how to build this file and finds that it needs a file named *allchaps.txt*, which is also missing as it was deleted at line 13. So **make** turns to the rule for this file and finally finds something it can work with. At this point a complete rebuild begins.

A better approach is to create the dependency that you actually want. Set up the makefile so that *thebook.txt* depends directly on the five chapters, as in the following example.

```
 1 # builds thebook.txt from the chapters using one rule
 2
 3 .SUFFIXES: .txt
 4
 5 CHAPTERS= chap1.txt chap2.txt chap3.txt chap4.txt chap5.txt
 6
 7 thebook.txt : $(CHAPTERS)
 8    cat $(CHAPTERS) >allchaps.txt
 9    toc allchaps.txt >toc.txt
10    cat toc.txt allchaps.txt >tocchaps.txt
11    paginate tocchaps.txt
12    idx tocchaps.txt >thebook.txt
13    paginate thebook.txt
14    rm -f toc.txt
15    rm -f allchaps.txt
16    rm -f tocchaps.txt
```

In this listing, a variable has been used to create a list of the chapters. Line 5 begins *CHAPTERS=* and sets the variable *$(CHAPTERS)* to the names of the five chapters. Later in the makefile, *$(CHAPTERS)* is used as shorthand for the list of files. This is a handy way of creating a list so that, if you later create a sixth chapter, you can just add it to the variable *CHAPTERS* and the makefile will run correctly and produce a new version of the book.

This listing includes only one rule and one set of dependencies. The whole book is built only if either *thebook.txt* does not exist or any of the chapters was modified since the last time *thebook.txt* was created. This is essentially the trick with **make** – getting the dependencies right. Another feature of **make** is that the subject of the first rule is its default target. In the above sample makefile,

```
make thebook.txt
```

and

```
make
```

are equivalent, because the rule for *thebook.txt* is the first one in the file.

Now suppose that you also want to create a printed hard copy of any chapter that changed. You can use **make** to do this, but you need to use a little trick.

What you want to do is print chapters if they have changed since they were last printed. The problem is that a printed copy of a chapter does not produce a file. **make** uses file dependencies to determine whether to execute the commands, but printing does not create a target file that can be checked. So, how can you check whether a file was printed since it was last modified? To do this, create a dummy file named *printed* whenever you print an updated chapter. Consider the listing below, which uses this trick. The example contains another problem that we'll look at in a moment, but first consider how **make** uses the dummy file.

```
1 # Prints out-of-date chapters
2
3 .SUFFIXES: .txt
4
5 CHAPTERS= chap1.txt chap2.txt chap3.txt chap4.txt chap5.txt
6
7 printed : $(CHAPTERS)
8    pr $(CHAPTERS) | lp
9    touch printed
```

The rule in line 7 checks whether a file named *printed* exists and, if necessary, creates it using the commands. First the chapters are printed (line 8) and then the file named *printed* is created or updated using **touch** (line 9). If a *printed* file doesn't exist, an empty one is created by the **touch** command; if one does exist, its modification date and time is updated. In this example, the file is used only as a marker to indicate when the chapters were printed. The next time **make printed** is run, *printed*'s date and time stamp will show that *printed* is up-to-date with respect to all the chapters.

The problem with this listing is that, if any chapter has been updated, all of them are printed. What we really want is to print only chapters that have been updated. The **make** utility includes a special macro variable that stands for 'all dependent files that have a later date and time stamp than the target file'. The macro is a dollar sign followed by a question mark ('$?'). If we rewrite the above listing using this macro it becomes:

```
1 # Prints out-of-date chapters
2
3 .SUFFIXES: .txt
4
5 CHAPTERS= chap1.txt chap2.txt chap3.txt chap4.txt chap5.txt
6
7 printed : $(CHAPTERS)
8    pr $? | lp
9    touch printed
```

If chapters 3 and 5 were modified since the last time the chapters were printed out, the **pr** command in line 8 is expanded to:

```
    pr chap3.txt chap5.txt | lp
```

The *printed* file is still ‘touched’ to indicate that all of the out-of-date chapters have now been printed. Notice the difference in the two problems addressed by **make**. In the first, we want to use all chapters to recreate the book if any chapter is updated. In the second, we want to print only updated chapters. The *$?* macro variable is perfect for the latter.

More than one **make** rule can be combined in one makefile, and the rules don’t have to be related. The two **make** rules that we’ve looked at so far are combined with two additional ones shown in the next listing. Using this makefile, you can type **make thebook.txt** to create the latest version of the book, **make printed** to print out updated chapters, **make chaps.tar.Z** to create a compressed **tar** archive of the chapters, or **make list** to list the component chapters. The third and fourth rules require some further explanation.

The rule for creating *chaps.tar.Z* (line 23) shows another of **make**’s interesting features. The first step is to remove the old version of the file using the command **-rm .chaps.tar.Z** (line 24). The hyphen in front of the **rm** command is a signal to the **make** utility that it should not quit if there’s an error. The normal behaviour of **make** is to exit if an error occurs. The first time that you try to create *chaps.tar.Z,* the file won’t exist, and so **rm** fails with an error. Normally this error would stop **make** from proceeding any further. However, the hyphen tells **make** that this is not a critical command and an error can be ignored.

The fourth rule (line 28) has a target (‘list’), but no dependencies. When this occurs in a makefile, no checking for out-of-date dependencies is done, and the associated commands are simply executed. Line 29 echoes the names of the input files that are processed by this makefile on the screen.

```
 1 # Builds thebook.txt from the chapter files.
 2 # Also prints, lists, or archives chapters.
 3
 4 .SUFFIXES: .txt
 5
 6 CHAPTERS= chap1.txt chap2.txt chap3.txt chap4.txt chap5.txt
```

```

thebook.txt : $(CHAPTERS)
   cat $(CHAPTERS) >allchaps.txt
   toc allchaps.txt >toc.txt
   cat toc.txt allchaps.txt >tocchaps.txt
   paginate tocchaps.txt
   idx tocchaps.txt >thebook.txt
   paginate thebook.txt
   rm -f toc.txt
   rm -f allchaps.txt
   rm -f tocchaps.txt

printed : $(CHAPTERS)
   pr $? | lp
   touch printed

chaps.tar.Z : $(CHAPTERS)
   -rm chaps.tar.Z
   tar cf chaps.tar $(CHAPTERS)
   compress chaps.tar

list :
   echo $(CHAPTERS)
```

As mentioned earlier, **make** uses the subject of the first rule as the default target. If you simply type **make** and press *Enter*, the rule for *thebook.txt* would be executed as the first one in the file. Most **make** files are arranged so that the main process appears as the first rule.

There is another of **make**'s features that's worth pointing out. A makefile can contain a command that executes **make**. Assume that you decide that, whenever you rebuild the book, you also want to update the archive *chaps.tar.Z*. One easy way to handle this is to add a new first rule in the **make** file as in the example below in lines 7 to 9. This first rule has no dependencies, so the commands are always executed. The first command runs **make** to create the book (line 8), and the second runs **make** to create the archive (line 9). This makefile can be started by typing **make** or **make bookarch**.

```
1 # Builds thebook.txt and prints, lists, or archives chapters.
2
3 .SUFFIXES: .txt
4
5 CHAPTERS= chap1.txt chap2.txt chap3.txt chap4.txt chap5.txt
6
7 bookarch:
8    make thebook.txt
```

```
   make chaps.tar.Z

thebook.txt : $(CHAPTERS)
   cat $(CHAPTERS) >allchaps.txt
   toc allchaps.txt >toc.txt
   cat toc.txt allchaps.txt >tocchaps.txt
   paginate tocchaps.txt
   idx tocchaps.txt >thebook.txt
   paginate thebook.txt
   rm -f toc.txt
   rm -f allchaps.txt
   rm -f tocchaps.txt

printed : $(CHAPTERS)
   pr $? | lp
   touch printed

chaps.tar.Z : $(CHAPTERS)
   -rm chaps.tar.Z
   tar cf chaps.tar $(CHAPTERS)
   compress chaps.tar

list :
   echo $(CHAPTERS)
```

There's a lot more to the **make** utility than I've covered here (many of the features I've omitted were originally designed to handle program development), but the examples I've given should show you how to use it as an administration tool.

Mo Budlong
President
KCSI (USA)

Administering RS/6000s with a Palm Pilot

System consoles are a necessary part of every server installation. While most system administration tasks can be performed from remote locations using network connections, some require direct physical connection to the server. However, ASCII terminals and graphical displays are bulky and it's impractical to install one for every server in large installation. Fortunately, owners of SP systems are able to control hundreds of servers using a single control workstation. This article discusses this innovative solution to the console problem, a solution that is already available from IBM.

SNAPP is a Web-based client/server application that assists in basic system administration tasks using a Palm Pilot handheld device as a console. When in use, the Palm Pilot cradle's serial port is connected to a serial port on the RS/6000 system.

The application itself consists of three parts:

- A Palm Pilot application, which includes an XML parser, a display component, and a server communication module.
- A SnappDaemon server (**snappd**) that runs on the RS/6000.
- A set of Perl scripts that are used to implement a number of the application's functions.

INSTATION

The application is available (with very limited support) from:

```
http://www.alphaworks.ibm.com/tech/snapp
```

No payment is required, but an on-line licence agreement must be signed. The application is downloaded as a single *snapp.tar* archive file, which should be extracted in the */usr* directory of the RS/6000 server. This results in the application being placed in directory */usr/local/snapp*. The tool is supported under AIX 4.3.3 with following OS file-sets installed:

- *bos.compat.links*
- *bos.acct*
- *perl.rte*.

The **snapdd** daemon can be started manually using the command:

```
/usr/local/snapp/bin/snappdrun
```

To start it automatically during system start-up, add the following line to */etc/inittab*:

```
    snappd:2:once:/usr/local/snapp/bin/snappdrun &> /dev/null 2>&1
```

The **snappd** daemon uses */dev/tty0* by default. If you'd like to use a different *tty*, invoke **snappd** with the *tty* number as its only command line option – **snappd 2** directs the server to user *tty* '2'.

The Palm Pilot client application is in file *snapp.prc*, which is located in the */usr/local/snapp/Pilot* directory. This should be transferred to a PC that has the Palm Pilot installer software on it and installed on the Palm Pilot. The OS version of Palm Pilot should be 3.0 or later.

The next step is to connect the Palm Pilot cradle's serial cable to the serial port that corresponds to the *tty* used by **snappd**.

OPERATION

The Palm Pilot SNAPP application's icon is a light bulb. When the icon is selected, an animation of two plugs trying to connect is shown. When a connection is made with the server, the animation shows two connected plugs, with bits passing in both directions.

The client then sends the *root* password authentication request to the server. After a successful authentication, the application's home page is transferred from the server to the client. This page contains a list of all functions that are defined in the SNAPP application. A button represents each function; when a button is selected, the following information is sent to the server:

- The name of the button
- The name of the associated Perl script
- The value of any text fields filled in by the user.

This information is parsed by the server, which invokes the designated Perl script with the parameters supplied, processes the data produced by the script, formats the information into an XML document, and sends the document back to the client. The XML page, which contains both the layout information and the user interface element definitions, is displayed on the Palm Pilot.

Every SNAPP page includes three icons in the upper right-hand corner: 'back', 'home', and 'reload'. 'Back' and 'reload' are selfexplanatory, while ‘home’ takes you to SNAPP’s home page. The ‘reload’ button is particularly valuable when displaying statistical information, such as performance data, as reloading the page updates the data.

root password authentication is performed both during the SNAPP Palm Pilot application’s start-up and also when communication with the server is broken or the client is inactive for more than ten minutes.

FUNCTIONS

Network set-up

When a button labelled ‘Network Setup’ is selected from the SNAPP application’s home page, the following information about the first server network interface available is displayed:

- The name of the interface (part of the page title)
- The host name
- The IP address
- The subnet mask
- The default gateway’s IP address
- The domain and name server’s IP address.

The user can select the *TestPing*, *Modify*, or *Clear* buttons. If an error occurs after a *Modify* (or *Clear* then *Modify*), the problem fields are masked with asterisks (‘*’). The user then has a choice of either correcting the problem or cancelling the changes. A *TestPing* should be performed once the desired configuration is properly set. The application pings the host name, its IP address, the gateway’s IP address, and the name server’s IP address. The results of the checks are reported on the next screen.

Note that, in order to define a brand new system using this method, you must first make sure that SNAPP is installed on the system and that the **snappd** daemon is operational. A common way of doing this is to install the system using a **mksysb** image of a server that has **snappd** started during system boot via */etc/innittab*.

Apache status

Selecting this button displays the status of an Apache Web server running on the target server. If the server is currently active, the user is provided with a button that can be used to stop it; if the server is stopped, the interface allows it to be started.

System statistics

This page displays statistics for CPU, memory, and disk usage. For multi-processor systems, CPU usage for all CPUs is reported. The memory report displays the total memory currently in use. Disk usage reports the disk space on all disks connected to the system that's either available or in use. The report also includes system configuration details, such as model number, CPU type, OS level, number of processors, and physical memory installed. Press the 'Reload' to refresh the information displayed.

System shut-down and reboot

These pages display buttons for shutting down and rebooting the server. A cautionary confirmation of these actions is required before they are executed.

EXTENSIBILITY

SNAPP's design enables easy extensibility. The only parts that need to be added or changed to extend it are Perl scripts located in the */usr/local/snapp/scripts/AIX* directory. For instance, to add functionality for Apache control, the following changes are made:

- Add a new menu selection to the *Hoe.pl* script
- Add four script files, that implement the additional functionality:
  - *ApacheStatus.pl*
  - *ApacheStart.pl*
  - *ApacheStop.pl*
  - *ApacheRestart.pl*.

It's worth noting that fewer than 100 lines of Perl are needed to implement this functionality.

REFERENCES

1 *SNAPP: An XML-based Palm Pilot Interface to RS/6000 Administration*, IBM Server Group RS/6000 Partners in Development.

2 SNAPP's *readme* file.

Alex Polak
System Engineer
APS (Israel)

Killing idle connections

On a busy system, idle connections can be nuisance. The problem is that, on the one hand, the system administrator cannot expect users to utilize their logons one hundred percent of the time, while, on the other, too many idle connections may prevent additional users from logging onto the system. This places the burden of deciding on the definition of an idle connection for a given system on system administrators. This will depend on factors such as the type of user that may log on to the system, the type of activity that may be carried out, etc. **timeout.sh** is a shell script that automates this task and purges idle connections.

TIMEOUT.SH

```
##########################################################################
# Name        : timeout.sh
#
# Description : The script processes users' idle connection time and
#               kills sessions that have been idle for longer than
#               the time input.
#
```

```
# Input       1 Time in minutes against which idle connection time
#               is compared. This parameter is mandatory.
#
#             2 Logfile name. This parameter is optional.
#
# Notes       1 Usage: timeout.sh <t=time> l=<log file name>
#
#             2 The script contains following functions:
#               o main
#               o InitializeVariables
#               o ParseCommandLine
#               o ValidateArgumentValues
#               o InitializeLogFile
#               o ProcessConnectionIdleTime
#               o ProcessExit
#
#             3 If a log file is not specified, the script appends
#               or initializes log output to /tmp/timeout.log.
#
#             4 If a log file is specified, the script initializes
#               the file (if it doesn't exist) or appends log output
#               to the file (if it exists).
#
#             5 Ideally this script should run in cron under root
#               account. An example entry into crontab is as follows:
#
#               15,30,45,59 * * * * timeout.sh > /tmp/timeout.err 2>&1
######################################################################
######################################################################
# Name        : InitializeVariables
#
# Description : Initializes module variables.
######################################################################
InitializeVariables ()
{
# date and time
DATETIME=`date "+%d/%m/%y at %H:%M:%S"`

# working variables
USER=         # connection owner
PID=          # connecting process id
TERMINAL=     # connecting terminal
TIMEOUT=      # timeout value
LOG_FILE=     # log file

# temporary files
USER_LIST="/tmp/timeout_$$_user.lis"

# default log file
DEFAULT_LOG_FILE="/tmp/timeout.log"
# exit code
SEC=0
FEC=1

# return code
TRUE=0
FALSE=1

# message prefixes
ERROR="timeout.sh:ERROR:"
INFO="timeout.sh:INFO:"

# messages
USAGE="Usage\:timeout.sh t=\<time\> l=\<log file name\>"
DUP_ARG="Duplicate argument"
INVALID_ARGC="Wrong number of argument counts"
INVALID_ARG_TYPE="\${ARG_TYPE}, is an invalid argument type"
LOG_NOT_INITIALIZED="Failed to initialize log file \${LOG_FILE}"
INVALID_TIMEOUT_VALUE="\${TIMEOUT}, is invalid timeout value"
}
######################################################################
# Name     : DisplayMessage
#
# Overview : Displays message
#
# Input    : 1 Message type (E = Error, I = Informative)
#            2 Error code as defined in DefineMessages ().
######################################################################
DisplayMessage ()
{
MESSAGE_TYPE=$1
MESSAGE_TEXT=`eval echo $2`

clear
if ["${MESSAGE_TYPE}" = "E"]
then
    echo "`eval echo ${ERROR}`${MESSAGE_TEXT}\c"
else
    echo "`eval echo ${INFO}`${MESSAGE_TEXT}\c"
fi
return ${TRUE}
}
######################################################################
# Name     : ValidateArgumentValues
#
# Overview : Validates argument values
#
# Returns  : $TRUE or $FALSE
#
# Notes    1 The following command line parameters are examined:
```

```
#            o TIMEOUT
#            o LOG_FILE
######################################################################
ValidateArgumentValues ()
{
# validate timeout value
if [$TIMEOUT -lt 1]
then
    DisplayMessage E "${INVALID_TIMEOUT_VALUE}"
    return $FALSE
fi

# log file existence
if ["${LOG_FILE}" = ""]
then
    # no logfile name has been supplied default it
    LOG_FILE="${DEFAULT_LOG_FILE}"
fi
return $TRUE
}
######################################################################
# Name        : InitializeLogFile
#
# Description : Initializes the log file
#
# Returns     : $TRUE or $FALSE
######################################################################
InitializeLogFile ()
{
if [! -f ${LOG_FILE}]
then
  # initialize the log file
  (> ${LOG_FILE}) > /dev/null 2>&1
  if [$? -ne 0]
  then
      DisplayMessage E "${LOG_NOT_INITIALIZED}"
      return $FALSE
  fi
  echo "Log file for idle connection termination on ${DATETIME} " >> \
       ${LOG_FILE}
  echo "=========================================================" >>\
       ${LOG_FILE}
fi
return $TRUE
}

######################################################################
# Name     : ParseCommandLine
#
# Overview : Parses the command line parameters
```

```
#
# Returns  : $TRUE or  $FALSE
#
# Notes    1 The following command line parameters are expected:
#            t=<time>
#            l=<logfile name>
######################################################################
ParseCommandLine ()
{
# establish argument count
if [${ARGC} -lt 1 -o ${ARGC} -gt 2]
then
    DisplayMessage E "${INVALID_ARGC}"
    DisplayMessage I "${USAGE}"
    return $FALSE
fi

# process arguments
INDEX=1
while [! $INDEX -gt $ARGC]
do
    # extract next argument line
    ARG_LINE=`echo "${ARGV}" | cut -d' ' -f${INDEX}`

    # extract argument type
    ARG_TYPE=`echo ${ARG_LINE} | cut -c1-1`
    case "${ARG_TYPE}" in
        l|t) # check for duplicate argument type
             if ["${ARG_TYPE}" = "t" -a "${TIMEOUT}" != ""]
             then
                  DisplayMessage E "${DUP_ARG}" ;
             elif ["${ARG_TYPE}" = "t" -a  "${TIMEOUT}" = ""]
             then
                  # store this argument value
                  TIMEOUT=`echo "${ARG_LINE}" | cut -d'=' -f2` ;
             elif ["${ARG_TYPE}" = "l" -a  "${LOG_FILE}" != ""]
             then
                  DisplayMessage E "${DUP_ARG}" ;
             elif ["${ARG_TYPE}" = "l" -a "${LOG_FILE}" = ""]
             then
                  # store this argument value
                  LOG_FILE=`echo "${ARG_LINE}" | cut -d'=' -f2`;
             fi ;;
          *) DisplayMessage E "${INVALID_ARG_TYPE}" ;
             DisplayMessage I "${USAGE}" ;
             return $FALSE ;;
    esac
    INDEX=`expr $INDEX + 1`
done
return $TRUE
```

```
}
###############################################################################
# Name        : ProcessConnectionIdleTime
#
# Description : Processes all users' idle connection time
###############################################################################
ProcessConnectionIdleTime ()
{
# create a list of users currently connected
who -u > ${USER_LIST}

# process each connection
cat ${USER_LIST} | while read LINE
do
    USER=`echo "${LINE}" | awk {'print $1'}`
    TERMINAL=`echo "${LINE}" | awk {'print $2'}`
    IDLE=`echo "${LINE}" | awk {'print $6'}`
    PID=`echo "${LINE}" | awk {'print $7'}`

    # for IDLE = "." connection is processing
    if ["${IDLE}" = "."]
    then
        continue
    fi

    # convert idle time to minutes
    HOURS=`echo "${IDLE}" | cut -d':' -f1`
    MINS=`echo "${IDLE}" | cut -d':' -f2`

    # calculate idle elapsed time
    IDLE_ELAPSED_TIME=`expr \($HOURS \* 60 \) + $MINS`

    # compare idle elapsed time to timeout
    if [$IDLE_ELAPSED_TIME -lt $TIMEOUT]
    then
        continue
    fi

    # send message to timed-out connection
    echo "
        timeout.sh:INFO:Terminating connection after $TIMEOUT
        ➤  minute(s)
         idle
        " >> /dev/$TERMINAL

    # terminate the connection
    KillProcess
done
}
###############################################################################
```

```
# Name        : KillProcess
#
# Description : Kills a process
#
# Notes       1 The function uses the following global variables:
#               o USER
#               o TERMINAL
#               o PID
#               o TIMEOUT
######################################################################
KillProcess ()
{
# Write connection details to log file
DATETIME=`date "+%d/%m/%y at %H:%M:%S"`
echo "
    timeout.sh:INFO:Terminating ${USER}'s connection(Process
    ➤  Id=$PID) to
    /dev/${TERMINAL} after $TIMEOUT minute(s) idle on $DATETIME
    " >> $LOG_FILE
# try signal -1 first
kill -1 $PID
if [$? -eq 0]
then
    echo "
        timeout.sh:INFO:Successfully Terminated connection
        " >> ${LOG_FILE}
    return $TRUE
fi

# try signal -9
kill  -9  $PID
if [$? -eq 0]
then
    echo "
        timeout.sh:INFO:Successfully Terminated connection
        " >> ${LOG_FILE}
else
    echo "
        timeout.sh:INFO:Failed to Terminate connection
        " >> ${LOG_FILE}
    return $FALSE
fi
return $TRUE
}
######################################################################
# Name        : ProcessExit
#
# Description : Performs a graceful exit.
#
# Input       : Exit code.
```

```
###############################################################################
ProcessExit ()
{
EXIT_CODE="$1"

# remove temporary file
rm -f ${USER_LIST}
exit  $EXIT_CODE
}
###############################################################################
# Name        : main
#
# Description : Calls all other functions.
###############################################################################
main ()
{
InitializeVariables
if ! ParseCommandLine
then
    ProcessExit $FEC
fi
if ! ValidateArgumentValues
then
    ProcessExit $FEC
fi
if ! InitializeLogFile
then
    ProcessExit $FEC
fi
ProcessConnectionIdleTime
ProcessExit $SEC
}

# store command line argument count and values
ARGC="$#"
ARGV="$@"

# invoke main
main
```

Arif Zaman
DBA/AIX Administrator
High-Tech Software (UK)

grep extended

INTRODUCTION

grepe.sh (**grep** extended) is a shell script that extends the power of the normal **grep** command. It would be nice if **grep** would let you view the search pattern in a window containing required number of lines of text above and below the search string.

This would be particularly useful for viewing a source files. **grepe.sh** is a shell script that implements this using **grep** and **csplit**.

One item that requires customization is *DEFAULT_PRINTER*, which is defined in the first section of code.

Note the use of the continuation character, '➤', in the code below to indicate a formatting line break that's not present in either the original source code or the code that can be downloaded from Xephon's Web site (*http://www.xephon.com/aixupdate.html*).

GREPE.SH

```
##########################################################################
# Name     : grepe.sh (grep extended)
#
# Overview : The script allows the user to search for a string in a
#            text file and view the string in a window that displays
#            a number of lines above and below the string.
#
# Usage:   : grepe.sh s=<string> f=<filename> l=<lines> i=<Y|N>
##########################################################################
##########################################################################
# Name     : InitializeVariables
#
# Overview : Initializes all required variables.
##########################################################################
InitializeVariables  ()
{
    # temporary files
    M_INPUT_FILE="/tmp/grepe_$$_input.dat"   # input file line numbers
    OUTPUT_FILE="/tmp/grepe_$$_output.dat"   # result file
    TEMP_FILE="/tmp/grepe_$$.tmp"            # temporary file
    TEMP_FILE_1="/tmp/grepe_$$_1.tmp"        # temporary file
    TEMP_FILE_2="/tmp/grepe_$$_2.tmp"        # temporary file
```

```
SPLIT_FILES="/tmp/grepe_$$_csplit"       # files created by csplit
REQ_SPLIT_FILE="/tmp/grepe_$$_csplit01"  # split text file
# Input parameters
P_STRING=""       # search string
P_INPUT_FILE=""   # input file
P_LINES=""        # lines to display above and below string
P_IGNORE_CASE=""  # ignore case
# Define default printer
DEFAULT_PRINTER="eiffel"
# Exit codes
SEC=0                      # Success exit code
FEC=99                     # Failure exit code
# Return codes
TRUE=0
FALSE=1
# Escape sequences
ESC="\0033["
RVON= [7m                  # Reverse video on
RVOFF= [27m                # Reverse video off
BOLDON= [1m                # Bold on
BOLDOFF= [22m              # Bold off
BON= [5m                   # Blinking on
BOFF= [25m                 # Blinking off
SLEEP_DURATION=4           # Seconds for sleep command
ERROR="${RVON}${BON}grepe.sh:ERROR:${BOFF}"
INFO="${RVON}grepe.sh:INFO:${BOFF}"
# Messages
WORKING="Working...${RVOFF}"
INTERRUPT="Program interrupted! Quitting${RVOFF}"
INVALID_ENTRY="Invalid entry${RVOFF}"
INVALID_ARGC="Invalid argument count${RVOFF}"
DUP_ARG="\${ARG_TYPE}, is a duplicate argument${RVOFF}"
INVALID_ARG_TYPE="\${ARG_TYPE}, is an invalid argument${RVOFF}"
USAGE="Usage:grepe.sh s=\<string\> f=\<full file name\>
➤  l=\<lines to display\> i=\<y\|n\>${RVOFF}"
OS_ERROR="\${ERR_MSG}${RVOFF}"
INVALID_STRING="Invalid search string${RVOFF}"
INPUT_FILE_REQ="Must provide a text file name${RVOFF}"
INPUT_FILE_NOT_FOUND="File \${P_INPUT_FILE} does not exist
➤  ${RVOFF}"
INPUT_FILE_NOT_READABLE="File \${P_INPUT_FILE} is not readable by
➤  user${RVOFF}"
LINES_REQ="Must provide number of lines to display${RVOFF}"
INVALID_LINES="\${P_LINES} is an invalid number of lines to
➤  display${RVOFF}"
LINES_NOT_NUMERIC="Number of lines to display, \${P_LINES}, must be
➤  numeric${RVOFF}"
PRINT_OK="Successfully submitted the print job${RVOFF}"
PRINT_NOT_OK="Failed to submit the print job${RVOFF}"
CASE_REQ="Must enter a value for argument ignore case\(i\)${RVOFF}"
INVALID_VALUE="Invalid value for argument ignore case\(i\)${RVOFF}"
```

```
    # Define signals
    SIGHUP=1    ;  export SIGHUP     # Session disconnected
    SIGINT=2    ;  export SIGINT     # Ctrl-c
    SIGTERM=15  ;  export SIGTERM    # kill command
    SIGTSTP=18  ;  export SIGTSTP    # Interactive stop (ctrl-z)
}
#######################################################################
# Name     : FormatUnderscores
#
# Overview : Assigns the right number of underscore characters ('=')
#            to the variable UNDERSCORE (used in conjunction with a
#            header).
#
# Input    : Line containing the header
#######################################################################
FormatUnderscores ()
{
    LINE="$1"
    UNDERSCORE=
    IND=1
    NO_CHARS=`echo "$LINE" | wc -c`
    NO_CHARS=`expr $NO_CHARS - 1`
    while [ "$IND" -le "$NO_CHARS" ]
    do
        UNDERSCORE="${UNDERSCORE}="
        IND=`expr $IND + 1`
    done
}
#######################################################################
# Name     : HandleInterrupt
#
# Overview : Calls ProcessExit.
#######################################################################
HandleInterrupt ()
{
    DisplayMessage I "${INTERRUPT}"
    ProcessExit $FEC
}
#######################################################################
# Name     : MoveCursor
#
# Overview : Moves cursor to location (Y, X).
#
# Input    : Y and X coordinates.
#######################################################################
MoveCursor ()
{
    trap "HandleInterrupt " $SIGINT $SIGTERM $SIGHUP $SIGTSTP
    YCOR=$1
    XCOR=$2
    echo "${ESC}${YCOR};${XCOR}H"
```

```
}
#######################################################################
# Name     : DisplayMessage
#
# Overview : The function displays message
#
# Input    : 1. Message type (E = Error, I = Information)
#            2. Error code is defined in DefineMessages ()
#            3. Message to be acknowledged flag  (Y = yes, N = no).
#######################################################################
DisplayMessage ()
{
    trap "HandleInterrupt " $SIGINT $SIGTERM $SIGHUP $SIGTSTP
    MESSAGE_TYPE=$1
    MESSAGE_TEXT=`eval echo $2`
    ACKNOWLEDGE_FLAG="$3"
    if [ "${ACKNOWLEDGE_FLAG}" = "" ]
    then
        ACKNOWLEDGE_FLAG="Y"
    fi
    MoveCursor 24 1
    if [ "${MESSAGE_TYPE}" = "E" ]
    then
        echo "`eval echo ${ERROR}`${MESSAGE_TEXT}\c"
    else
        echo "`eval echo ${INFO}`${MESSAGE_TEXT}\c"
    fi
    # examine message acknowledge flag
    if [ "${ACKNOWLEDGE_FLAG}" = "Y" ]
    then
        read DUMMY
    else
        sleep ${SLEEP_DURATION}
    fi
    return ${TRUE}
}
#######################################################################
# Name     : ProcessExit
#
# Overview : Makes a graceful exit.
#
# Input    : Exit code.
#######################################################################
ProcessExit ()
{
    # assign parameter
    EXIT_CODE="$1"
    # remove files
    rm -f ${M_INPUT_FILE}
    rm -f ${OUTPUT_FILE}
    rm -f ${TEMP_FILE}
    rm -f ${TEMP_FILE_1}
    rm -f ${TEMP_FILE_2}
    rm -f ${SPLIT_FILES}*
    exit ${EXIT_CODE}
}
##################################################################################
# Name     : ProcessInputFile
#
# Overview : Searches the input file for the specified string.
#
# Returns  : $TRUE or $FALSE.
##################################################################################
ProcessInputFile ()
{
    DisplayMessage I "${WORKING}" N
    # Initialize local variables
    FIRST_LINE_NO=1   # First line no in input file
    LAST_LINE_NO=0    # Last line no in input file
    LINE_NO=1         # Working variable
    STRING_LINE_NO=0  # Number of line containing string
    LINES_ABOVE=0     # Number of first line to be displayed
    LINES_BELOW=0     # Number of last line to be displayed
    NO_OCCURRENCE=0   # Number of occurrences of search string
    DATETIME=`date "+%d/%m/%Y at %H:%M:%S"`
    STRING_FOUND=N    # Indicates whether search was successful
    # Initialize output file
    HEADER="Search Results on ${DATETIME}"
    FormatUnderscores "${HEADER}"
    echo "            ${HEADER}"      > ${OUTPUT_FILE}
    echo "            ${UNDERSCORE}" >> ${OUTPUT_FILE}
    echo "                 Parameter details" >> ${OUTPUT_FILE}
    echo "                 =================" >> ${OUTPUT_FILE}
    HEADER="Search string=${P_STRING} File=${P_INPUT_FILE}
    ➤  Lines=${P_LINES} Ignore case=${P_IGNORE_CASE}"
    FormatUnderscores "${HEADER}"
    echo "          ${HEADER}"       >> ${OUTPUT_FILE}
    echo "          ${UNDERSCORE}\n" >> ${OUTPUT_FILE}
    # Initialize modified input file
    > ${M_INPUT_FILE}
    # Number records in input file
    cat ${P_INPUT_FILE} | while read LINE
    do
        echo "${LINE}" | sed s/^/${LINE_NO}:/ >> ${M_INPUT_FILE}
        LINE_NO=`expr $LINE_NO + 1`
    done
    LAST_LINE_NO=`expr $LINE_NO - 1`
    if [ "${P_IGNORE_CASE}" = "Y" -o "${P_IGNORE_CASE}" = "y" ]
    then
        grep -i -n "${P_STRING}" ${P_INPUT_FILE} > ${TEMP_FILE}
    else
        grep -n "${P_STRING}" ${P_INPUT_FILE} > ${TEMP_FILE}
    fi
    cat ${TEMP_FILE} | while read LINE
    do
        NO_OCCURRENCE=`expr $NO_OCCURRENCE + 1`
        STRING_FOUND=Y
        STRING_LINE_NO=`echo "${LINE}" | cut -d':' -f1`
        LINE="${LINE}<---"
        sed s/^${STRING_LINE_NO}:.*/"${LINE}"/ ${M_INPUT_FILE}
        ➤  > ${TEMP_FILE_2}
        cat  ${TEMP_FILE_2}  >  ${M_INPUT_FILE}
        LINES_ABOVE=`expr ${STRING_LINE_NO} - ${P_LINES}`
        if [ $LINES_ABOVE -le 0 ]
        then
            LINES_ABOVE=${FIRST_LINE_NO}
        fi
        LINES_BELOW=`expr ${STRING_LINE_NO} + ${P_LINES}`
        if [ $LINES_BELOW -gt $LAST_LINE_NO ]
        then
            LINES_BELOW=${LAST_LINE_NO}
        fi
        if [ $LINES_BELOW -eq $STRING_LINE_NO ]
        then
            echo "${LINE}" > ${TEMP_FILE_1}
        elif [ $LINES_BELOW -eq $LAST_LINE_NO ]
        then
            csplit -s -f ${SPLIT_FILES} ${M_INPUT_FILE}
            ➤  '/'${LINES_ABOVE}:'/'
            cat ${REQ_SPLIT_FILE}  > ${TEMP_FILE_1}
        else
            csplit -s -f ${SPLIT_FILES} ${M_INPUT_FILE}
            ➤  '/'${LINES_ABOVE}:'/' '/'${LINES_BELOW}:'/+1'
            cat ${REQ_SPLIT_FILE} > ${TEMP_FILE_1}
        fi
        # concatenate result into output file
        HEADER="Occurrence No=${NO_OCCURRENCE}"
        FormatUnderscores "${HEADER}"
        echo "${HEADER}"     >> ${OUTPUT_FILE}
        echo "${UNDERSCORE}" >> ${OUTPUT_FILE}
        cat ${TEMP_FILE_1}   >> ${OUTPUT_FILE}
        echo "\n"            >> ${OUTPUT_FILE}
    done
    # check STRING_FOUND flag
    if [ "${STRING_FOUND}" = "N" ]
    then
        echo "\n\n       No occurrences were found" >> ${OUTPUT_FILE}
    fi
}
#######################################################################
# Name     : ParseCommandLine
#
# Overview : Parses the command line parameters.
```

```
#
# Returns  : $TRUE or $FALSE
#
# Notes    : 1. The following command-line parameters are expected:
#               o s=<value>      assigned to P_STRING
#               o f=<full name>              P_INPUT_FILE
#               o l=<value>                  P_LINES
#               o i=<value>                  P_IGNORE_CASE
######################################################################
ParseCommandLine ()
{
    trap "HandleInterrupt " $SIGINT $SIGTERM $SIGHUP $SIGTSTP
    # establish argument count
    if [ ${ARGC} -lt 3 -o ${ARGC} -gt 4 ]
    then
        DisplayMessage E "${INVALID_ARGC}" "N"
        DisplayMessage E "${USAGE}"
        return $FALSE
    fi
    # process arguments
    INDEX=1
    while [ ! $INDEX -gt $ARGC ]
    do
        ARG_LINE=`echo "${ARGV}" | cut -d'|' -f${INDEX}`
        ARG_TYPE=`echo ${ARG_LINE} | cut -d'=' -f1 | tr "a-z" "A-Z"`
        ARG_TYPE_LEN=`expr "${ARG_TYPE}" : '.*'`
        ARG_TYPE_LEN=`expr ${ARG_TYPE_LEN} + 2`
        case "${ARG_TYPE}" in
            S|F|L|I)
                if [ "${ARG_TYPE}" = "S" -a "${P_STRING}" != "" ]
                then
                    DisplayMessage E "${DUP_ARG}" "N" ;
                    DisplayMessage I "${USAGE}" "Y" ;
                    return $FALSE ;
                elif [ "${ARG_TYPE}" = "S" -a "${P_STRING}" = "" ]
                then
                    P_STRING=`echo "${ARG_LINE}" | cut -c
                    ➤  ${ARG_TYPE_LEN}-` ;
                elif [ "${ARG_TYPE}" = "F" -a "${P_INPUT_FILE}" != "" ]
                then
                    DisplayMessage E "${DUP_ARG}" "N" ;
                    DisplayMessage I "${USAGE}" "Y" ;
                    return $FALSE ;
                elif [ "${ARG_TYPE}" = "F" -a "${P_INPUT_FILE}" = "" ]
                then
                    P_INPUT_FILE=`echo "${ARG_LINE}" | cut -c
                    ➤  ${ARG_TYPE_LEN}-` ;
                elif [ "${ARG_TYPE}" = "L" -a "${P_LINES}" != "" ]
                then
                    DisplayMessage E "${DUP_ARG}"  "N" ;
                    DisplayMessage I "${USAGE}" "Y" ;
                    return $FALSE ;
                elif [ "${ARG_TYPE}" = "L" -a "${P_LINES}" = "" ]
                then
                    P_LINES=`echo "${ARG_LINE}" | cut -c
                    ➤  ${ARG_TYPE_LEN}-` ;
                elif [ "${ARG_TYPE}" = "I" -a "${P_IGNORE_CASE}"
                    ➤  != "" ]
                then
                    DisplayMessage E "${DUP_ARG}" "N" ;
                    DisplayMessage I "${USAGE}" "Y" ;
                    return $FALSE ;
                elif [ "${ARG_TYPE}" = "I" -a "${P_IGNORE_CASE}"
                    ➤  = ""  ]
                then
                    # store this argument value
                    P_IGNORE_CASE=`echo "${ARG_LINE}" | cut -c
                    ➤  ${ARG_TYPE_LEN}-` ;
                fi ;;
                 * ) DisplayMessage E "${INVALID_ARG_TYPE}" "N" ;
                     DisplayMessage I "${USAGE}" "Y" ;
                     return $FALSE ;;
        esac
        INDEX=`expr $INDEX + 1`
    done
    if [ "${P_STRING}" = "" ]
    then
        DisplayMessage E "${INVALID_STRING}" "Y"
        return $FALSE
    fi
    # P_INPUT_FILE
    if [ "${P_INPUT_FILE}" = "" ]
    then
        DisplayMessage E "${INPUT_FILE_REQ}" "Y"
        return $FALSE
    elif  [ ! -f "${P_INPUT_FILE}" ]
    then
        DisplayMessage E "${INPUT_FILE_NOT_FOUND}" "Y"
        return $FALSE
    elif  [ ! -r "${P_INPUT_FILE}" ]
    then
        DisplayMessage E "${INPUT_FILE_NOT_READABLE}" "Y"
        return $FALSE
    fi
    # P_LINES
    if [ "${P_LINES}" = "" ]
    then
        DisplayMessage E "${LINES_REQ}" "Y"
        return $FALSE
    elif ! [ `expr ${P_LINES} + 1 2> /dev/null` ]
    then
        DisplayMessage E "${LINES_NOT_NUMERIC}"
        return $FALSE
    elif [ ${P_LINES} -lt 0 ]
    then
        DisplayMessage E "${INVALID_LINES}" "Y"
        return $TRUE
    fi
    # P_IGNORE_CASE
    case "${P_IGNORE_CASE}" in
             "" ) if [ $ARGC -eq 4 ]
                  then
                      DisplayMessage E "${CASE_REQ}" ;
                      return $FALSE ;
                  else
                      P_IGNORE_CASE=Y ;
                  fi ;;
        y|Y|n|N ) : ;;
          * ) DisplayMessage E "${INVALID_VALUE}" ;
              return $FALSE ;;
    esac
}
######################################################################
# Name     : PrintResultFile
#
# Overview : Prints the named file.
#
# Input    : Name of the file to be printed.
######################################################################
PrintResultFile ()
{
    FILE_TO_BE_PRINTED=$1
    while true
    do
        clear
        echo "Do you wish to print the output file (Y/N)?:\c"
        read REPLY
        case $REPLY in
            n|N ) return $TRUE ;;
            y|Y ) break ;;
              * ) DisplayMessage E "${INVALID_ENTRY}" ;;
        esac
    done
    while true
    do
        clear
        echo "Enter printer name for lp command(q to quit):\c"
        read PRINTER
        case $PRINTER in
            "" ) : ;;
            q|Q) break ;;
             * ) lp -d$PRINTER ${FILE_TO_BE_PRINTED} > ${TEMP_FILE_1}
                ➤  2>&1 ;
```

```
                 if [ $? -eq 0 ]
                 then
                     DisplayMessage I "${PRINT_OK}" ;
                     break ;
                 else
                     DisplayMessage E "${PRINT_NOT_OK}" "N" ;
                     # Remove * from the error file in order to
                     # process the error correctly.
                     sed s/\*// ${TEMP_FILE_1} > ${TEMP_FILE_2} ;
                     ERR_MSG=`cat ${TEMP_FILE_2}` ;
                     DisplayMessage E "${OS_ERROR}" ;
                 fi ;;
        esac
    done
}
######################################################################
# Name     : DisplayResults
#
# Overview : Displays the result file and allows the user to be able
#            to print the result file.
#
# Notes    : 1. The function calls the following functions:
#               o PrintResultFile
######################################################################
DisplayResults ()
{
    view ${OUTPUT_FILE}
    PrintResultFile
}
######################################################################
# Name     : main
#
# Overview : Implements the processing structure and invokes all other
#            functions.
#
# Notes    : 1. The function calls the following functions:
#               o InitializeVariables
#               o ParseCommandLine
#               o ProcessExit $FEC
#               o ProcessInputFile
#               o DisplayResults
######################################################################
main ()
{
    InitializeVariables
    # handle signals
    trap "HandleInterrupt " $SIGTSTP $SIGTERM $SIGHUP $SIGINT
    if ! ParseCommandLine
    then
        ProcessExit $FEC
    fi
    ProcessInputFile
    DisplayResults
    ProcessExit $SEC
}
# Assign command-line parameters for later processing
ARGC="$#"
ARG1="$1"
ARG2="$2"
ARG3="$3"
ARG4="$4"
ARGV="${ARG1}|${ARG2}|${ARG3}|${ARG4}"
# Invoke main
main
```

SAMPLE OUTPUT

```
            Search results on 13/02/2000 at 23:31:26
            ========================================
                 Parameter Details
                 =================
          Search string=nature File=/mi.proc Lines=3 Ignore case=Y
          =======================================================

Occurrence No=1
===============
88:i_mi_offer_type        IN varchar2,
89:i_mi_id                IN varchar2,
90:i_mi_status            IN varchar2,
91:i_mi_nature            IN varchar2,  ﬂ- search string
92:i_mi_type              IN varchar2,
93:i_mi_category          IN varchar2,
94:i_mi_level             IN varchar2,

Occurrence No=2
===============
115:i_lns_mi_pan          IN varchar2,
116:i_lns_row_num         IN varchar2,
117:i_lns_mi_maintainable IN varchar2,
118:o_mi_nature_status   OUT varchar2, ﬂ- search string
119:o_error_type         OUT varchar2,
120:o_error_code         OUT varchar2,
121:o_error_msg          OUT varchar2,

Occurrence No=3
===============
144:--
145:check_value varchar2(2);
146:p_userid    varchar2(32);
147:l_mi_nature_status  varchar2(1); ﬂ- search string
148:l_mi_warranty varchar2(2);
```

```
149:/*
150:* cursor declaration for mi status

Occurrence No=4
===============
155:WHERE  status_code = i_mi_status
156:AND    status_status = 'V';
157:/*
158:* cursor declaration for mi naturefl--  search  string
159:*/
160:CURSOR chk_mi_nature IS
161:SELECT 'x'
```

Arif Zaman
DBA/System Administrator
High-Tech Software (UK)

Mailto – an AIX Web server extension (part 2)

Last month we published an article containing a detailed description of Mailto, a Web server extension that does away with much of the hard work associated with handling Web-based forms. This month we publish the code for Mailto in its entirety. We conclude this article next month by publishing the program's on-line documentation.

MAILTO.C

```
/* Mailto.c / Mailing data filled in web forms / Pierre Croisetiere */
#include <errno.h>
#include <signal.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <time.h>
#define MAILPGM "/usr/sbin/sendmail -t -oi -oem"   /* sendmail invocation     */
#define TMPFILE "/tmp/Mailto.tmp"                  /* Temporary work file     */
#define LOGFILE "/usr/log/Mailto.log"              /* Log file absolute path  */
#define NSQFILE "/usr/log/Mailto.noseq"            /* Save file: _$NOSEQ fct  */
#define ONLNOTE 0                                  /* Online usage notes      */
#define HLPHTTP "http://www.server.name"           /* URL-serv to usage notes */
```

```
#define HLPHTML "/FullWebPath/Mailto.htm"          /* URL-file to usage notes */
#define DFLTSND "Web@Some.Net"                     /* Default sender email    */
#define DFLTSUB "No title."                        /* Default email subject   */
#define DOMCHCK 0                                  /* Referer domain check    */
#define DOMLIST \
{\
        ".domain.name.suffix",\
}                                                  /* List authorized domains */
#define DBGPATH "/debug/"                 /* PATHINFO invoking debug mode     */
#define HLPPATH "/help/"                  /* PATHINFO invoking help           */
#define NOHPATH "/nohelp/"                /* PATHINFO reporting no online hlp */
#define CGINAME "Mailto 3.0"              /* Program name and version number  */
#define CGIFONT "Arial,Geneva,sans-serif" /* Font for dynamic HTML pages      */
#define TIMNOPN 5                         /* Nbr of attempts to open NSQFILE  */
#define TIMWAIT 1                         /* Wait time between attempts       */
#define FDBUFLN 2048                      /* Fd file buffer size              */
#define MAXVARL 512                       /* Error msg: max size of var part  */
#define MAXFIXL 256                       /* Error msg: max size of fix part  */
#define MAXREFE 256                       /* Max size of referers full name   */
#define MAXDOML 96                        /* Max size of referers domain name */
#define MAXDGIT 10                        /* Nb of digits of NOSEQ in NSQFILE */
#define MAXLNSQ (MAXDOML+MAXDGIT+5)       /* Max line size of NSQFILE         */
#define MAXLINE 512                       /* Max value for _LINE              */
#define OPENREF '{'                       /* Reference initiator character    */
#define CLOSREF '}'                       /* Reference terminator character   */
#define ESC_REF '\\'                      /* Escape character                 */
#define PTR_REF 0x95                      /* Reference place holder character */
#define OPN_LST "{"                       /* Instances list opening char      */
#define SEP_LST " & "                     /* Instances list separator char    */
#define CLO_LST "}"                       /* Instances list closing char      */
#define END     ((struct CGI_PRM *)0)     /* Parameter list end pointer       */
#define END_REF ((struct REF_PRM *)0)     /* Reference list end pointer       */
#define END_DIR ((struct MAILTO_PRM *)0)  /* Instruction list end pointer     */
#define ALL     "_ALL"                    /* Generic field name (all fields)  */
#define syserr  (sys_errlist [errno])     /* UNIX system errors table         */
static char D_EMAIL[] = DFLTSND;          /* Dflt From: header               */
static char D_SUBJT[] = DFLTSUB;          /* Dflt Subject: header            */
static char D_LINE[]  = "80";             /* Dflt _LINE value                */
static char D_MARGN[] = "0";              /* Dflt _MARGIN value              */
static char D_MXGAP[] = "0";              /* Dflt _MAX_GAP value             */
static char D_SFNAM[] = " = ";            /* Dflt _SUFFIX_NAME value         */
static char D_SFVAL[] = "\n";             /* Dflt _SUFFIX_VALUE value        */
static char D_OMITT[] = "Empty/omitted";  /* Empty/omitted instr placeholder */
static char D_NOSTR[] = "";               /* Empty str when no default apply */
char DocUrl[MAXREFE], PagNam[MAXREFE], Dom_Name_Lst [] [MAXDOML+1] = DOMLIST;
int  Nb_Dom = sizeof(Dom_Name_Lst)/sizeof(Dom_Name_Lst [0]);
extern char *sys_errlist[];
/* Instruction codes definition */
#define  MANDATORY      0x00000001   /* Field attribute instruction */
#define  NUMERICAL      0x00000002   /* Field attribute instruction */
#define  LOWERCASE      0x00000003   /* Field attribute instruction */
```

```
#define  UPPERCASE      0x00000004   /* Field attribute instruction */
#define  INVISIBLE      0x00000005   /* Field attribute instruction */
#define  INVISIBLE_COND 0x00000006   /* Field attribute instruction */
#define  TO             0x00000007   /* Mailing instruction         */
#define  TO_1X1         0x00000008   /* Mailing instruction         */
#define  FROM           0x00000009   /* Mailing instruction         */
#define  SUBJECT        0x0000000a   /* Mailing instruction         */
#define  CC             0x0000000b   /* Mailing instruction         */
#define  BCC            0x0000000c   /* Mailing instruction         */
#define  REPLY_TO       0x0000000d   /* Mailing instruction         */
#define  LINE           0x0000000e   /* Formatting instruction      */
#define  MARGIN         0x0000000f   /* Formatting instruction      */
#define  MAX_GAP        0x00000010   /* Formatting instruction      */
#define  SUFFIX_NAME    0x00000011   /* Formatting instruction      */
#define  SUFFIX_VALUE   0x00000012   /* Formatting instruction      */
#define  WRITE          0x00000013   /* Formatting instruction      */
#define  WRITE_COND     0x00000014   /* Formatting instruction      */
#define  XFR_URL        0x00000015   /* Control instruction         */
#define  IF             0x00000020   /* Control instruction         */
#define  AND            0x00000021   /* Control instruction         */
#define  OR             0x00000022   /* Control instruction         */
#define  ELSE_IF        0x00000023   /* Control instruction         */
#define  ELSE           0x00000024   /* Control instruction         */
#define  END_IF         0x00000025   /* Control instruction         */
/* Instruction characteristics: single-bit masks (16-bit frame in MAILTO_PRM)
   DIR_WRIT: Message building         WRITE WRITE_COND;
   DIR_EDIT: Allow \t \n              WRITE WRITE_COND SUBJECT
                                      SUFFIX_NAME SUFFIX_VALUE;
   DIR_REFS: Allow mult reference     WRITE WRITE_COND SUBJECT;
   DIR_FROM: Allow {_FROM} as Email   BCC CC REPLY_TO;
   DIR_MULT: Chain all instances      SUBJECT TO TO_1X1 BCC CC REPLY_TO;
   DIR_UNIQ: Honor only last instance FROM LINE MARGIN MAX_GAP
                                      SUFFIX_NAME SUFFIX_VALUE XFR_URL;
   DIR_ATTR: Field attribute          MANDATORY NUMERICAL LOWERCASE UPPERCASE
                                      INVISIBLE INVISIBLE_COND;
   DIR_LOGI: Test cond proposal       IF AND OR;
   DIR_SECT: Close IF-Block section   ELSE_IF ELSE END_IF;
   DIR_EXIS: Inst found in form       Set dynamically when inst is found; */
#define  DIR_WRIT 0x00000004
#define  DIR_EDIT 0x00000008
#define  DIR_REFS 0x00000010
#define  DIR_FROM 0x00000020
#define  DIR_MULT 0x00000040
#define  DIR_UNIQ 0x00000080
#define  DIR_ATTR 0x00000100
#define  DIR_LOGI 0x00000200
#define  DIR_SECT 0x00000400
#define  DIR_EXIS 0x00010000
/* Field single-bit state masks (16-bit frame in CGI_PRM) */
#define  PRM_INVI 0x00000001 /* Invisible field/instruction       */
#define  PRM_WRIT 0x00000100 /* Unconditional print instr         */
```

```
#define  PRM_NULL 0x00000200 /* All occurrences are empty         */
#define  PRM_SYST 0x00000400 /* IF-Block inst or system var       */
#define  PRM_IGNO 0x00000800 /* Ignore instr (inactive IF branch) */
/* CGI env variables set by web server */
char *Cgi_AUTH_TYPE,         *Cgi_CONTENT_LENGTH,  *Cgi_CONTENT_TYPE;
char *Cgi_GATEWAY_INTERFACE, *Cgi_HTTP_ACCEPT,     *Cgi_HTTP_CONNECTION;
char *Cgi_HTTP_HOST,         *Cgi_HTTP_PRAGMA,     *Cgi_HTTP_REFERER;
char *Cgi_HTTP_USER_AGENT,   *Cgi_HTTPS,           *Cgi_PATH_INFO;
char *Cgi_PATH_TRANSLATED,   *Cgi_QUERY_STRING,    *Cgi_REMOTE_ADDR;
char *Cgi_REMOTE_HOST,       *Cgi_REMOTE_IDENT,    *Cgi_REMOTE_USER;
char *Cgi_REQUEST_METHOD,    *Cgi_SCRIPT_NAME,     *Cgi_SERVER_NAME;
char *Cgi_SERVER_PORT,       *Cgi_SERVER_PROTOCOL, *Cgi_SERVER_SOFTWARE;
char *Cgi_SERVER_URL;
#define Cgi_ENV_INIT \
{\
   "AUTH_TYPE",         &Cgi_AUTH_TYPE,\
   "CONTENT_LENGTH",    &Cgi_CONTENT_LENGTH,\
   "CONTENT_TYPE",      &Cgi_CONTENT_TYPE,\
   "GATEWAY_INTERFACE", &Cgi_GATEWAY_INTERFACE,\
   "HTTP_ACCEPT",       &Cgi_HTTP_ACCEPT,\
   "HTTP_CONNECTION",   &Cgi_HTTP_CONNECTION,\
   "HTTP_HOST",         &Cgi_HTTP_HOST,\
   "HTTP_PRAGMA",       &Cgi_HTTP_PRAGMA,\
   "HTTP_REFERER",      &Cgi_HTTP_REFERER,\
   "HTTP_USER_AGENT",   &Cgi_HTTP_USER_AGENT,\
   "HTTPS",             &Cgi_HTTPS,\
   "PATH_INFO",         &Cgi_PATH_INFO,\
   "PATH_TRANSLATED",   &Cgi_PATH_TRANSLATED,\
   "QUERY_STRING",      &Cgi_QUERY_STRING,\
   "REMOTE_ADDR",       &Cgi_REMOTE_ADDR,\
   "REMOTE_HOST",       &Cgi_REMOTE_HOST,\
   "REMOTE_IDENT",      &Cgi_REMOTE_IDENT,\
   "REMOTE_USER",       &Cgi_REMOTE_USER,\
   "REQUEST_METHOD",    &Cgi_REQUEST_METHOD,\
   "SCRIPT_NAME",       &Cgi_SCRIPT_NAME,\
   "SERVER_NAME",       &Cgi_SERVER_NAME,\
   "SERVER_PORT",       &Cgi_SERVER_PORT,\
   "SERVER_PROTOCOL",   &Cgi_SERVER_PROTOCOL,\
   "SERVER_SOFTWARE",   &Cgi_SERVER_SOFTWARE,\
   "SERVER_URL",        &Cgi_SERVER_URL\
}
struct CGI_ENV_VAR { char NAM[30], **VAL; } CgiEnv [] = Cgi_ENV_INIT;
int    Nb_CGI_ENV_VAR=sizeof(CgiEnv)/sizeof(CgiEnv [0]), CgiCntLen;
char   *Cgi_Stdin_Stream;
/* CGI parameter structure definition */
struct CGI_PRM { char *name, *value, *prec; int lnam, lval; unsigned int state;
                 struct CGI_PRM *next, *nxtoc, *ptroc; struct REF_PRM *Refer; };
struct CGI_PRM Cgi_Dflt_FROM, Cgi_Dflt_SUBJECT;
struct CGI_PRM *FirstWrkPrm, *FirstUrlPrm , *FirstFrmPrm;
struct CGI_PRM *FirstPrm, **Cgi_Prm_Prec, *Sys_Refs();
int    Nbr_Prm, NbrUrlPrm, NbrFrmPrm, NbrWrkPrm;
int    Prm_Buf_Len, MaxPrmLen, MaxUrlPrmLen, MaxFrmPrmLen;
char   *Prm_Buf;
/* Mailto instruction set declaration and initialization */
int ValIntPrm(), Val_E_Mail(), ValXfrUrl(), Sav_As_Is(), No_Op();
struct MAILTO_PRM { char name[20], *value; int (*Validate_Fct)();
                    struct CGI_PRM *first, *last; unsigned int code, state; } \
\
   Dir_AND            = {"_AND",            D_NOSTR, No_Op,      END, END,\
                           AND,             DIR_LOGI },\
   Dir_BCC            = {"_BCC",            D_NOSTR, Val_E_Mail, END, END,\
                           BCC,             DIR_MULT | DIR_FROM },\
   Dir_CC             = {"_CC",             D_NOSTR, Val_E_Mail, END, END,\
                           CC,              DIR_MULT | DIR_FROM },\
   Dir_ELSE           = {"_ELSE",           D_NOSTR, No_Op,      END, END,\
                           ELSE,            DIR_SECT },\
   Dir_ELSE_IF        = {"_ELSE_IF",        D_NOSTR, No_Op,      END, END,\
                           ELSE_IF,         DIR_SECT },\
   Dir_END_IF         = {"_END_IF",         D_NOSTR, No_Op,      END, END,\
                           END_IF,          DIR_SECT },\
   Dir_FROM           = {"_FROM",           D_EMAIL, Val_E_Mail, END, END,\
                           FROM,            DIR_UNIQ },\
   Dir_IF             = {"_IF",             D_NOSTR, No_Op,      END, END,\
                           IF,              DIR_LOGI },\
   Dir_INVISIBLE      = {"_INVISIBLE",      D_NOSTR, No_Op,      END, END,\
                           INVISIBLE,       DIR_ATTR },\
   Dir_INVISIBLE_COND = {"_INVISIBLE_COND", D_NOSTR, No_Op,      END, END,\
                           INVISIBLE_COND,  DIR_ATTR },\
   Dir_LINE           = {"_LINE",           D_LINE,  ValIntPrm,  END, END,\
                           LINE,            DIR_UNIQ },\
   Dir_LOWERCASE      = {"_LOWERCASE",      D_NOSTR, No_Op,      END, END,\
                           LOWERCASE,       DIR_ATTR },\
   Dir_MANDATORY      = {"_MANDATORY",      D_NOSTR, No_Op,      END, END,\
                           MANDATORY,       DIR_ATTR },\
   Dir_MARGIN         = {"_MARGIN",         D_MARGN, ValIntPrm,  END, END,\
                           MARGIN,          DIR_UNIQ },\
   Dir_MAX_GAP        = {"_MAX_GAP",        D_MXGAP, ValIntPrm,  END, END,\
                           MAX_GAP,         DIR_UNIQ },\
   Dir_NUMERICAL      = {"_NUMERICAL",      D_NOSTR, No_Op,      END, END,\
                           NUMERICAL,       DIR_ATTR },\
   Dir_OR             = {"_OR",             D_NOSTR, No_Op,      END, END,\
                           OR,              DIR_LOGI },\
   Dir_REPLY_TO       = {"_REPLY_TO",       D_NOSTR, Val_E_Mail, END, END,\
                           REPLY_TO,        DIR_MULT | DIR_FROM },\
   Dir_SUBJECT        = {"_SUBJECT",        D_SUBJT, No_Op,      END, END,\
                           SUBJECT,         DIR_EDIT | DIR_REFS | DIR_MULT },\
   Dir_SUFFIX_NAME    = {"_SUFFIX_NAME",    D_SFNAM, Sav_As_Is,  END, END,\
                           SUFFIX_NAME,     DIR_EDIT | DIR_UNIQ },\
   Dir_SUFFIX_VALUE   = {"_SUFFIX_VALUE",   D_SFVAL, Sav_As_Is,  END, END,\
                           SUFFIX_VALUE,    DIR_EDIT | DIR_UNIQ },\
   Dir_TO             = {"_TO",             D_NOSTR, Val_E_Mail, END, END,\
                           TO,              DIR_MULT },\
   Dir_TO_1X1         = {"_TO_1X1",         D_NOSTR, Val_E_Mail, END, END,\
                           TO_1X1,          DIR_MULT },\
   Dir_UPPERCASE      = {"_UPPERCASE",      D_NOSTR, No_Op,      END, END,\
                           UPPERCASE,       DIR_ATTR },\
   Dir_WRITE          = {"_WRITE",          D_NOSTR, No_Op,      END, END,\
                           WRITE,           DIR_WRIT | DIR_EDIT | DIR_REFS },\
   Dir_WRITE_COND     = {"_WRITE_COND",     D_NOSTR, No_Op,      END, END,\
                           WRITE_COND,      DIR_WRIT | DIR_EDIT | DIR_REFS },\
   Dir_XFR_URL        = {"_XFR_URL",        D_NOSTR, ValXfrUrl,  END, END,\
                           XFR_URL,         DIR_UNIQ };
struct MAILTO_PRM *Get_Dir_Ptr ();
/* Mailto instruction set sorted table */
struct MAILTO_PRM *Mailto_Prm [] = \
{\
   &Dir_AND            ,\
   &Dir_BCC            ,\
   &Dir_CC             ,\
   &Dir_ELSE           ,\
   &Dir_ELSE_IF        ,\
   &Dir_END_IF         ,\
   &Dir_FROM           ,\
   &Dir_IF             ,\
   &Dir_INVISIBLE      ,\
   &Dir_INVISIBLE_COND ,\
   &Dir_LINE           ,\
   &Dir_LOWERCASE      ,\
   &Dir_MANDATORY      ,\
   &Dir_MARGIN         ,\
   &Dir_MAX_GAP        ,\
   &Dir_NUMERICAL      ,\
   &Dir_OR             ,\
   &Dir_REPLY_TO       ,\
   &Dir_SUBJECT        ,\
   &Dir_SUFFIX_NAME    ,\
   &Dir_SUFFIX_VALUE   ,\
   &Dir_TO             ,\
   &Dir_TO_1X1         ,\
   &Dir_UPPERCASE      ,\
   &Dir_WRITE          ,\
   &Dir_WRITE_COND     ,\
   &Dir_XFR_URL        ,\
};
int Nb_Mailto_Prm = sizeof(Mailto_Prm)/sizeof(Mailto_Prm[0]);
/* Sorted table of system vars for refs on _SUBJECT _WRITE and _WRITE_COND
   For each entry, Sys_Var.value must be initialized to an empty string */
static char _$DATE[50]="", _$DD[3]="", _$HRS[3]="", _$MIN[3]="";
static char _$MM[3]="", _$MONTH[10]="", _$NOSEQ[MAXDGIT+1]="", _$SEC[3]="";
static char _$TIME[9]="", _$WKDAY[10]="", _$YEAR[5]="", _$YRDAY[4]="";
static char _$YY[3] ="";
int    Sys_Dates(), Sys_Refer(), Sys_Noseq();
char   *Noseq_Fd_Buf;
```

```
struct FD_REC       { int Fd, Bsiz, Balan, LasRl; char *Buf, *Pos; }\
       Noseq_File = { -1, FDBUFLN, 0, 0, NULL, NULL };
struct SYS_VAR { char *name, *value; struct CGI_PRM *PrmLoc; int (*SetVal)(); }\
       Sys_Var[] =\
{\
   "_$DATE",    _$DATE,    END, Sys_Dates, \
   "_$DD",      _$DD,      END, Sys_Dates, \
   "_$HRS",     _$HRS,     END, Sys_Dates, \
   "_$MIN",     _$MIN,     END, Sys_Dates, \
   "_$MM",      _$MM,      END, Sys_Dates, \
   "_$MONTH",   _$MONTH,   END, Sys_Dates, \
   "_$NOSEQ",   _$NOSEQ,   END, Sys_Noseq, \
   "_$REFERER",  D_NOSTR,  END, Sys_Refer, \
   "_$SEC",     _$SEC,     END, Sys_Dates, \
   "_$TIME",    _$TIME,    END, Sys_Dates, \
   "_$WKDAY",   _$WKDAY,   END, Sys_Dates, \
   "_$YEAR",    _$YEAR,    END, Sys_Dates, \
   "_$YRDAY",   _$YRDAY,   END, Sys_Dates, \
   "_$YY",      _$YY,      END, Sys_Dates, \
};
int Nb_Sys_Var = sizeof(Sys_Var)/sizeof(Sys_Var[0]);
/* Logical operators table for _IF, _AND, _OR _RLSE_IF */
#define  Is_LT 0x00000001  /* Operator flag: strictly less than    */
#define  Is_EQ 0x00000002  /* Operator flag: strictly equal to     */
#define  Is_GT 0x00000004  /* Operator flag: strictly greater than */
struct LOG_OPR { char name[5]; unsigned int cond; }\
       Log_Opr[] =\
{\
   ".LT.", Is_LT,         \
   ".LE.", Is_LT | Is_EQ, \
   ".EQ.", Is_EQ,         \
   ".GE.", Is_GT | Is_EQ, \
   ".GT.", Is_GT,         \
   ".NE.", Is_LT | Is_GT  \
};
int Nb_Opr = sizeof(Log_Opr)/sizeof(Log_Opr[0]);
/* Reference structure definition */
struct REF_PRM { struct CGI_PRM *targ; struct REF_PRM *next; };
struct REF_PRM *Ref_First, **Ref_Last;
int    NbrRef;
/* Miscellaneous variables and declarations */
FILE *Fp;
int  debug, Html_Header_Flag, Error_Flag;
int  Margin_Size, Line_Size, MaxGap, NbrChr, TabPos, Nbr_Mult;
char Force_Eol_Mark, *Suffix_Nam, *Suffix_Val, *MarginPtr;
char margin[MAXLINE+2], line[MAXLINE+2], WK[2*MAXVARL+MAXFIXL], datbuf[80];
char *WkDays[] = { "Sunday", "Monday", "Tuesday", "Wednesday",\
                   "Thursday", "Friday", "Saturday" };
char *ymonth[] = { "January",   "February", "March",    "April",
                   "May",       "June",     "July",     "August",
                   "September", "October",  "November", "December" };
```

```
char MrgSubj[] = "\n ";
char hlppath[] = HLPPATH;
char dbgpath[] = DBGPATH;
char UrlDsc[]  = "URL (QUERY_STRING)";
char FrmDsc[]  = "FORM (STDIN)";
char WrkDsc[]  = "WORK (Internal)";
char freefmt[] = "<B>Free_Mem: %d bytes, type %s</B><BR>\n";
char E_popen[] = "popen() system error";
char E_fork[]  = "fork() system error";
char *fdgets(), *datetime();
void GetCgiEnv(), PrtCgiEnv(), GetPrm(), PrtPrm(), IniCgiPrm();
void Html_Header(), Html_Trailer(), Free_Mem(), Redirect(), stgdump();
unsigned char xlat [256] = /* ISO-8859-1 Xlat tbl to blank unprintable chars */
{
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x0a, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
   0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f,
   0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
   0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f,
   0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47,
   0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f,
   0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57,
   0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f,
   0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67,
   0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f,
   0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77,
   0x78, 0x79, 0x7a, 0x7b, 0x7c, 0x7d, 0x7e, 0x7f,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7,
   0xa8, 0xa9, 0xaa, 0xab, 0xac, 0xad, 0xae, 0xaf,
   0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7,
   0xb8, 0xb9, 0xba, 0xbb, 0xbc, 0xbd, 0xbe, 0xbf,
   0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7,
   0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf,
   0xd0, 0xd1, 0xd2, 0xd3, 0xd4, 0xd5, 0xd6, 0xd7,
   0xd8, 0xd9, 0xda, 0xdb, 0xdc, 0xdd, 0xde, 0xdf,
   0xe0, 0xe1, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6, 0xe7,
   0xe8, 0xe9, 0xea, 0xeb, 0xec, 0xed, 0xee, 0xef,
   0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7,
   0xf8, 0xf9, 0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xff  };
unsigned char To_Upper [256] = /* Upper case xlat table */
{
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
   0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f,
   0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
   0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f,
   0x40, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47,
   0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f,
   0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57,
   0x58, 0x59, 0x5a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f,
   0x60, 0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47,
   0x48, 0x49, 0x4a, 0x4b, 0x4c, 0x4d, 0x4e, 0x4f,
   0x50, 0x51, 0x52, 0x53, 0x54, 0x55, 0x56, 0x57,
   0x58, 0x59, 0x5a, 0x7b, 0x7c, 0x7d, 0x7e, 0x7f,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7,
   0xa8, 0xa9, 0xaa, 0xab, 0xac, 0xad, 0xae, 0xaf,
   0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7,
   0xb8, 0xb9, 0xba, 0xbb, 0xbc, 0xbd, 0xbe, 0xbf,
   0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7,
   0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf,
   0xd0, 0xd1, 0xd2, 0xd3, 0xd4, 0xd5, 0xd6, 0xd7,
   0xd8, 0xd9, 0xda, 0xdb, 0xdc, 0xdd, 0xde, 0xdf,
   0xc0, 0xc1, 0xc2, 0xc3, 0xc4, 0xc5, 0xc6, 0xc7,
   0xc8, 0xc9, 0xca, 0xcb, 0xcc, 0xcd, 0xce, 0xcf,
   0xd0, 0xd1, 0xd2, 0xd3, 0xd4, 0xd5, 0xd6, 0xf7,
   0xd8, 0xd9, 0xda, 0xdb, 0xdc, 0xdd, 0xde, 0xff  };
unsigned char To_Lower [256] = /* Lower case xlat table */
{
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x21, 0x22, 0x23, 0x24, 0x25, 0x26, 0x27,
   0x28, 0x29, 0x2a, 0x2b, 0x2c, 0x2d, 0x2e, 0x2f,
   0x30, 0x31, 0x32, 0x33, 0x34, 0x35, 0x36, 0x37,
   0x38, 0x39, 0x3a, 0x3b, 0x3c, 0x3d, 0x3e, 0x3f,
   0x40, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67,
   0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f,
   0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77,
   0x78, 0x79, 0x7a, 0x5b, 0x5c, 0x5d, 0x5e, 0x5f,
   0x60, 0x61, 0x62, 0x63, 0x64, 0x65, 0x66, 0x67,
   0x68, 0x69, 0x6a, 0x6b, 0x6c, 0x6d, 0x6e, 0x6f,
   0x70, 0x71, 0x72, 0x73, 0x74, 0x75, 0x76, 0x77,
   0x78, 0x79, 0x7a, 0x7b, 0x7c, 0x7d, 0x7e, 0x7f,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20,
   0x20, 0xa1, 0xa2, 0xa3, 0xa4, 0xa5, 0xa6, 0xa7,
   0xa8, 0xa9, 0xaa, 0xab, 0xac, 0xad, 0xae, 0xaf,
   0xb0, 0xb1, 0xb2, 0xb3, 0xb4, 0xb5, 0xb6, 0xb7,
   0xb8, 0xb9, 0xba, 0xbb, 0xbc, 0xbd, 0xbe, 0xbf,
   0xe0, 0xe1, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6, 0xe7,
   0xe8, 0xe9, 0xea, 0xeb, 0xec, 0xed, 0xee, 0xef,
   0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xd7,
   0xf8, 0xf9, 0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xdf,
   0xe0, 0xe1, 0xe2, 0xe3, 0xe4, 0xe5, 0xe6, 0xe7,
   0xe8, 0xe9, 0xea, 0xeb, 0xec, 0xed, 0xee, 0xef,
   0xf0, 0xf1, 0xf2, 0xf3, 0xf4, 0xf5, 0xf6, 0xf7,
   0xf8, 0xf9, 0xfa, 0xfb, 0xfc, 0xfd, 0xfe, 0xff  };
unsigned char hextbl [256] = /* hex char to binary xlat table */
{
   0, 0, 0, 0, 0, 0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0, 0, 0, 0, 0, 0, 0,0,0,0,0,0,0,0,0,0,0,1,2,3,4,5,6,7,8,9,0,0,0,0,0,0,
   0,10,11,12,13,14,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0,10,11,12,13,14,15,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0, 0, 0, 0, 0, 0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0, 0, 0, 0, 0, 0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0, 0, 0, 0, 0, 0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
   0, 0, 0, 0, 0, 0, 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0  };
unsigned char Mail_Chr_Tbl [256] = /* Email address character class */
{
   0,    2, 2   , 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2,
   2,    2, 2   , 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2,
   0x20, 1, 0x22, 1, 1, 1, 1, 1, 2, 2, 1, 1, 2, 1, 0x2e, 1,
   1,    1, 1   , 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1, 2,    1,
   0x40, 1, 1   , 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,    1,
   1,    1, 1   , 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 1,    1,
   1,    1, 1   , 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,    1,
   1,    1, 1   , 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,    2,
   2,    2, 2,    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2,
   2,    2, 2,    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2,
   2,    2, 2,    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2,
   2,    2, 2,    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2,
   2,    2, 2,    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2,
   2,    2, 2,    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2,
   2,    2, 2,    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2,
   2,    2, 2,    2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,    2  };
   main (argc, argv)
   int argc;
   char *argv[];
{
   unsigned int code, state, All;
   int  i, j, k, child, nhlln, hlpln, dbgln, fnd, AllEmpty, empty;
   char c, *name, *value, *occur, *targ, *pfx;
   struct CGI_PRM *Prm, *Prm_a, *Prm_b; struct MAILTO_PRM *Directive;
/* Micellaneous checks and inits / Get Cgi Env variables */
   debug=Html_Header_Flag=Error_Flag = 0;
   Nbr_Prm=NbrUrlPrm=NbrFrmPrm=NbrWrkPrm=NbrRef = 0;
   MaxPrmLen=MaxUrlPrmLen=MaxFrmPrmLen=Nbr_Mult = 0;
   FirstPrm=FirstWrkPrm=FirstUrlPrm=FirstFrmPrm = END;
   Ref_First = END_REF;
   Ref_Last  = &Ref_First;
   Cgi_Stdin_Stream=Prm_Buf = Noseq_Fd_Buf = NULL;
   nhlln=strlen(NOHPATH);
   hlppath[(hlpln=strlen(hlppath))-1] = '\0';
   dbgpath[(dbgln=strlen(dbgpath))-1] = '\0';
   GetCgiEnv ();
   if ( sizeof(unsigned int) < 4 )
        Err ( "Error: unsigned int < 32 bits on this machine", NULL,1);
   for ( i=1; i<Nb_Mailto_Prm; i++ )
       { if ( strcmp (Mailto_Prm[i-1]->name,Mailto_Prm[i]->name) >= 0 )
              Err ( "Mailto_Prm instruction table not sorted", NULL,1); }
   for ( i=1; i<Nb_Sys_Var; i++ )
       { if ( strcmp (Sys_Var[i-1].name,Sys_Var[i].name) >= 0 )
              Err ( "Sys_Var system variables table not sorted", NULL,1); }
/* if needed set help/debug mode & chk if pgm called from an author domain */
   if ( ONLNOTE==1 ) sprintf (DocUrl,"%s%s",HLPHTTP,HLPHTML);
   else sprintf(DocUrl,"http://%s%s%s",Cgi_SERVER_NAME,Cgi_SCRIPT_NAME,NOHPATH);
   if      (Cgi_PATH_INFO == NULL );
   else if (strcmp (Cgi_PATH_INFO,hlppath)==0)       Redirect (DocUrl);
   else if (strncmp(Cgi_PATH_INFO,HLPPATH,hlpln)==0) Redirect (DocUrl);
   else if (strncmp(Cgi_PATH_INFO,NOHPATH,nhlln)==0) NoHelp ();
   else if (strcmp (Cgi_PATH_INFO,dbgpath)==0)       debug = 1;
   else if (strncmp(Cgi_PATH_INFO,DBGPATH,dbgln)==0) debug = 2;
   if ( Chk_HTTP_REFERER () ) {
        sprintf ( WK, "Program called from an unauthorised domain: \"%.*s\"",
                  MAXVARL, Cgi_HTTP_REFERER );
        Log (WK); Err (WK, NULL,1); }
   sprintf (WK, "Http_Referer: \"%.*s\"", MAXVARL, Cgi_HTTP_REFERER); Log (WK);
   PrtCgiEnv (argc, argv);
/* Extract from QUERY_STRING all CGI form parms transmited on URL */
   if ( Cgi_REQUEST_METHOD == NULL ) Err("Invalid REQUEST_METHOD value",NULL,2);
   if ( Cgi_CONTENT_LENGTH == NULL ) CgiCntLen = 0;
   else CgiCntLen = atoi (Cgi_CONTENT_LENGTH);
   if ( Cgi_QUERY_STRING != NULL )
      {
        GetPrm (Cgi_QUERY_STRING,&NbrUrlPrm,&FirstUrlPrm,&MaxUrlPrmLen,UrlDsc);
        PrtPrm (NbrUrlPrm,FirstUrlPrm,UrlDsc);
        Nbr_Prm = NbrUrlPrm; MaxPrmLen = MaxUrlPrmLen; FirstPrm = FirstUrlPrm; }
/* Extract CGI form parms laying on stdin
   Under REQUEST_METHOD=POST URL encoded parms are discarded */
   if ( strcmp ( Cgi_REQUEST_METHOD, "POST" ) == 0 )
      {
        if ( strcmp(Cgi_CONTENT_TYPE,"application/x-www-form-urlencoded") != 0 )
             Err ("CONTENT_TYPE not application/x-www-form-urlencoded",NULL,2);
        Cgi_Stdin_Stream = (char *) malloc ( CgiCntLen+1 );
        if ( fread ( Cgi_Stdin_Stream, 1, CgiCntLen, stdin) != CgiCntLen)
             Err ("CONTENT_LENGTH does not match stdin data size",NULL,1);
        Cgi_Stdin_Stream [ CgiCntLen ] = '\0';
        GetPrm (Cgi_Stdin_Stream,&NbrFrmPrm,&FirstFrmPrm,&MaxFrmPrmLen,FrmDsc);
        PrtPrm (NbrFrmPrm,FirstFrmPrm,FrmDsc);
        Nbr_Prm = NbrFrmPrm; MaxPrmLen = MaxFrmPrmLen; FirstPrm = FirstFrmPrm; }
   else if ( strcmp ( Cgi_REQUEST_METHOD, "GET" ) != 0 )
        Err ("REQUEST_METHOD is invalid",NULL,2);
/* - Scanning CGI form parms, detect, validate and process Mailto instructions
   - Retain last occurrence of single instance instruction
   - Chain occurrences of the same instruction when status allows many instances
   - Chain references on applicable statements
   - Apply attributes to all occurrences of target fields
   - Replace \n sequence by new line char in applicable statements */
   if ( FirstPrm == END ) Err ("No data available for processing",NULL,2);
   Cgi_Prm_Prec = &FirstWrkPrm;
   Prm_Buf = (char*) malloc (Prm_Buf_Len=MaxPrmLen+512+1);
   for ( Prm_a=FirstPrm; Prm_a != END; Prm_a=Prm_a->next )
     {
       name = Prm_a->name; value = Prm_a->value;
       if ( *name != '_' || (Prm_a->state & PRM_SYST) ) continue;
       for ( j=1, i=0; i<Nb_Mailto_Prm; i++ )
             if ( (j=strcmp(name, Mailto_Prm[i]->name)) <= 0 ) break;
       if ( j != 0 ) {
            sprintf (WK, "Invalid instruction: \"%.*s\"", MAXVARL, name);
            Err (WK, "INSTRSET",0); continue; }
       Directive = Mailto_Prm[i];
       code      = Directive->code;
       state     = Directive->state;
       if ( code == IF ) { Val_IF_Struct (Prm_a, Directive); continue; }
       if ( (state & (DIR_LOGI | DIR_SECT)) ) {
            BLIF_Err (Prm_a,"Instruction invalid outside IF-block"); continue; }
       if ( (state & DIR_WRIT) ) Prm_a->state |= PRM_WRIT;
       else Prm_a->state |= PRM_INVI;
       if ( (state & DIR_ATTR) ) {
            All=0;
            if ( strcmp (value, ALL) == 0 ) {
                 if   ( code != MANDATORY ) All=1;
                 else { Err("_ALL invalid on _MANDATORY",ALL,0); continue; } }
            else if ( *value == '_' ) {
                 for ( fnd=0, j=1, i=0; j>0 && i<Nb_Mailto_Prm; i++ ) {
                       if ( (j=strcmp (value, Mailto_Prm[i]->name)) == 0
                       && ( code == NUMERICAL || Mailto_Prm[i]->code != FROM) )
                          {
                            sprintf (WK,"%s not valid on %s",name,value);
                            Err (WK, name,0); fnd=1; } }
                 if ( fnd ) continue;
                 if ( j ) {
                      sprintf (WK, "%s applied on invalid instruction %.*s",
                                name, MAXVARL, value );
                      Err (WK, name, 0); fnd=1; continue; } }
            if ( Prm_a->state & PRM_IGNO ) continue;
            fnd=0; AllEmpty=1;
            for ( Prm_b=FirstPrm; Prm_b!=END; Prm_b=Prm_b->next )
```

```
              {
                occur = Prm_b->name;
                if ( All
                &&   (*occur!='_'||(code!=NUMERICAL&&strcmp(occur,"_FROM")==0))
                ||   strcmp(value,occur) == 0 )
                   {
                     fnd=1; targ=Prm_b->value; AllEmpty&=(empty=Is_Blank(targ));
                     if ( code==MANDATORY ) continue;
                     if ( code==INVISIBLE ) {Prm_b->state|=PRM_INVI; continue;}
                     if ( code==INVISIBLE_COND )
                        {
                          if ( empty ) Prm_b->state|=PRM_INVI; continue; }
                     if ( code==NUMERICAL ) { Val_Num (occur,targ); continue; }
                     if ( code==LOWERCASE ) { Set_Min (targ); continue; }
                     if ( code==UPPERCASE ) { Set_Maj (targ); continue; } } }
            if ( code==MANDATORY && ( fnd == 0 || AllEmpty ) )
               {
                 sprintf ( WK, "Mandatory field not completed: \"%.*s\"",
                           MAXVARL, value );
                 Err (WK, NULL,-1); }
            continue; }
        if ( state & DIR_EDIT ) Cnv_Edit (value);
        (*(Directive->Validate_Fct)) (Prm_a, Directive);
        if ( state & DIR_REFS ) Val_Refs (Prm_a, Directive);
        if ( (state & DIR_MULT) && *value && (Prm_a->state & PRM_IGNO) == 0 )
           {
            if    (Directive->first == END) Directive->first = Prm_a;
            else  (Directive->last)->nxtoc = Prm_a;
            Directive->last = Prm_a; Nbr_Mult++; } }
/* Ascertain presence of mandatory instruction _TO (_TO_1X1)
   Discard duplicates within _TO/_TO_1X1. Set effective value of ref {_FROM}
   Convert to numerical _LINE, _MARGIN & _MAX_GAP. Initialize margin buffer */
   if ( Dir_TO.first == END && Dir_TO_1X1.first == END )
        Err ( "At least one _TO or _TO_1X1 statement is mandatory", "_TO",0);
   Delete_Dup ( Dir_TO.first );        Delete_Dup ( Dir_TO_1X1.first );
   Delete_Dup ( Dir_CC.first );        Delete_Dup ( Dir_BCC.first );
   Delete_Dup ( Dir_REPLY_TO.first );
   Verif_Dir_Subject ( &Dir_SUBJECT );
   if ( Error_Flag != 0 ) Err (NULL, NULL, Error_Flag);
   for (Prm=Dir_FROM.first; Prm!=END; Prm=Prm->ptroc) Prm->value=Dir_FROM.value;
   if ( (Dir_MAX_GAP.state & DIR_EXIS) == 0 ) Dir_MAX_GAP.value=Dir_LINE.value;
   Line_Size   = atoi (Dir_LINE.value);     Suffix_Nam = Dir_SUFFIX_NAME.value;
   Margin_Size = atoi (Dir_MARGIN.value);   Suffix_Val = Dir_SUFFIX_VALUE.value;
   MaxGap      = atoi (Dir_MAX_GAP.value);
   margin[0]='\n'; for (i=1; i<=Margin_Size; i++) margin[i]=' '; margin[i]='\0';
   if ( debug==2 ) { PrtPrm ( Nbr_Prm,   FirstPrm,    CGINAME );
                     PrtPrm ( NbrWrkPrm, FirstWrkPrm, WrkDsc ); }
   Prt_Web_Dir ();
/* Formatting email on a pipe to sendmail or on screen if debug mode is on */
   if ( debug == 0 )
      {
         Force_Eol_Mark = ' ';
         if      ( (child=fork()) < 0 ) { Log (E_fork); Err (E_fork,NULL,1); }
         else if ( child == 0 ) goto fin ;
         if ( Dir_TO.first != END )
            {
              if ( (Fp = popen (MAILPGM,"w")) == NULL )
                 { Log ( E_popen ); Free_Mem(); exit(0); }
              for ( pfx="To: ", Prm=Dir_TO.first; Prm != END; Prm=Prm->nxtoc )
                  { fprintf ( Fp, "%s%s", pfx, Prm->value ); pfx = ",\n "; }
              fprintf (Fp,"\n"); Msg_Envelope (); Msg_Body (); pclose (Fp); }
         if ( Dir_TO_1X1.first != END)
            {
              for ( Prm=Dir_TO_1X1.first; Prm != END; Prm=Prm->nxtoc )
                  {
                    if ( (Fp = popen (MAILPGM,"w")) == NULL )
                       { Log ( E_popen ); Free_Mem(); exit(0); }
                    fprintf ( Fp, "To: %s\n", Prm->value );
                    Msg_Envelope (); Msg_Body (); pclose (Fp); } }
         Free_Mem(); exit (0); }
   else
      {
        Html_Header(); Fp = stdout; Force_Eol_Mark = '\\';
        Msg_Destinataire ();
        printf ("<B>E-mail headers and message body:</B></FONT><TT><PRE>\n");
        Msg_Envelope (); Msg_Body ();
        printf ("</PRE></TT><FONT FACE=\"%s\">\n<B>Next URL: ", CGIFONT );
        if ( ! Is_Blank (value=Dir_XFR_URL.value) )
           { printf ("<A HREF=\"%s\">%s</A>", value, value); }
        else printf ("undefined, Mailto will display message shown below" );
        printf ("<BR><BR><B>Debug Mode: <I>Email not sent</I></B><BR>\n"); }
/* End of processing / Print completion message or redirect to XFR_URL page */
   fin:
   if ( !debug && !Is_Blank(Dir_XFR_URL.value) ) Redirect(Dir_XFR_URL.value);
   if ( debug != 1 || Is_Blank (Dir_XFR_URL.value) )
      {
        Html_Header();
        printf ("<CENTER><BR><BR><HR SIZE=5 WIDTH=95%%><BR>\n\n<FONT SIZE=+1>");
        printf ("<B>\nForm, as completed, has been sent to whom it may ");
        printf ("concern<BR>\nby electronic mail, using %s</B>\n<BR>", CGINAME);
        printf ("<BR>\n%s usage notes are available on the Net:\n", CGINAME);
        printf ("<A HREF=\"%s\" TARGET=\"_top\">\nclick here</A>", DocUrl);
        printf ("<BR><BR>\n</FONT>\n<HR SIZE=5 WIDTH=95%%><BR></CENTER>\n"); }
   Free_Mem(); Html_Trailer(); exit(0);
}
   Msg_Destinataire ()
{
   struct CGI_PRM *Prm; char *pfx, *sfx;
   printf ("<BR><B>E-mail addressee(s):</B>\n</FONT><TT><PRE>");
   if ( Dir_TO_1X1.first != END)
      {
        for ( Prm=Dir_TO_1X1.first; Prm != END; Prm=Prm->nxtoc )
```

```
            printf ("TO_1X1: %s\n", Prm->value);
        if  ( Dir_TO.first != END) printf ("\n"); }
   if ( Dir_TO.first != END )
      {
        for ( pfx="TO: ", sfx="", Prm=Dir_TO.first; Prm != END; Prm=Prm->nxtoc )
            { printf ( "%s%s", pfx, Prm->value ); pfx = ",\n "; sfx = "\n"; }
        printf (sfx); }
   printf ("</PRE></TT><FONT FACE=\"%s\">\n", CGIFONT ); return;
}
   Msg_Envelope ()
{
   struct CGI_PRM *Prm; int save;
/* Build msg envelope with usual email headers. Close section with a blank line.
   Setting NbrChr=0 enforces application of _MARGIN, _LINE and _MAX_GAP */
   NbrChr=-1; Msg ( "From: " ); Msg ( Dir_FROM.value ); Msg ( "\n" );
   save=Line_Size; if ( Line_Size < 10 ) Line_Size=10;
   NbrChr=0;  TabPos=-1; MarginPtr=MrgSubj; Msg ( "Subject: " );
   for ( Prm=Dir_SUBJECT.first; Prm!=END; Prm=Prm->nxtoc ) { Expand_Refs(Prm); }
   if ( NbrChr ) fprintf ( Fp, "%s", line );
   NbrChr=-1; Msg ("\n");
   Msg_Header ( "Reply-To: ",  Dir_REPLY_TO.first );
   Msg_Header ( "Cc: ",        Dir_CC.first );
   Msg_Header ( "Bcc: ",       Dir_BCC.first );
   Msg ( "Comments: AIX " ); Msg ( CGINAME ); Msg ( "\n" );
   Msg ( "Mime-Version: 1.0\n" );
   Msg ( "Content-Type: text/plain; charset=\"iso-8859-1\"\n" );
   Msg ( "Content-Transfer-Encoding: 8bit\n" );
   Line_Size=save; NbrChr=0; TabPos=-1; MarginPtr=margin; Msg ("\n"); return;
}
   Msg_Header ( Header, Mail_Prm )
   char *Header; struct CGI_PRM *Mail_Prm;
{
   char *pfx, *sfx;
   for ( pfx=Header, sfx=""; Mail_Prm != END; Mail_Prm=Mail_Prm->nxtoc )
       { Msg ( pfx ); Msg ( Mail_Prm->value ); pfx = ",\n "; sfx = "\n"; }
   Msg (sfx); return;
}
   Msg_Body ()
{
   struct CGI_PRM *Prm;
   for ( Prm=FirstPrm; Prm != END; Prm=Prm->next )
     {
       if (  Prm->state & PRM_INVI ) continue;
       if ( (Prm->state & PRM_WRIT) == 0 )
          {
            Msg (Prm->name ); Msg (Suffix_Nam);
            Msg (Prm->value); Msg (Suffix_Val); continue; }
       Expand_Refs ( Prm ); }
   if ( NbrChr ) Msg ("\n");
   return;
}
   Expand_Refs (Prm_Ecr)
   struct CGI_PRM *Prm_Ecr;
{
   char *targ, *value; struct CGI_PRM *Prm, *Second; struct REF_PRM *Ref;
   if ( (Ref=Prm_Ecr->Refer) == END_REF ) { Msg (Prm_Ecr->value); return; }
   for ( targ=value=Prm_Ecr->value; *targ; targ++ )
     {
       if ( *targ == PTR_REF )
          {
            *targ = '\0'; Msg (value); *targ = PTR_REF;
            Prm = Ref->targ; Second = Prm->nxtoc;
            if ( Second != END ) Msg (OPN_LST); Msg (Prm->value);
            while ( (Prm=Prm->nxtoc)!=END ) { Msg (SEP_LST); Msg (Prm->value); }
            if ( Second != END ) Msg (CLO_LST);
            Ref = Ref->next; value = targ+1; } }
   Msg (value); return;
}
   Msg (str)
   char *str;
{
   int ncr; char c, *p, *q, *s;
   if ( *str == '\0' ) return;
   if ( NbrChr < 0 ) { fprintf ( Fp, "%s", str ); return; }
   p=str; q=line+NbrChr;
   while ( *p )
     {
       if ( *p == '\t' ) { if ( TabPos < 0 ) TabPos=NbrChr; p++; continue; }
       if ( (*(q++) = *(p++)) == '\n' )
          {
            *(--q)='\0';
            if ( TabPos >= 0 && TabPos < Line_Size && NbrChr < Line_Size )
               {
                 s=line+TabPos; c=*s; *s='\0'; if ( c == '\0' ) c=' ';
                 s=line+Line_Size; *(s--)='\0';
                 if ( TabPos < NbrChr ) while ( *s=*(--q) ) s--;
                 while ( NbrChr++ <= Line_Size ) *(s--)=c; }
            fprintf ( Fp, "%s%s", line, MarginPtr );
            q=line; NbrChr=0; TabPos=-1; continue; }
       if ( ++NbrChr <= Line_Size ) continue;
       if ( TabPos >= 0 ) TabPos = Line_Size;
       s=q-2; --NbrChr; ncr = (MaxGap < NbrChr) ? MaxGap : NbrChr;
       while ( *s != ' ' && !ispunct(*s) && --ncr ) --s;
       if   ( ncr > 0 )
            {
              c=*(++s); *s='\0';
              fprintf ( Fp, "%s%c%s", line, Force_Eol_Mark, MarginPtr );
              *s=c; *q='\0'; NbrChr=0; q=line;
              while ( *(q++)=*(s++) ) ++NbrChr; q--; }
       else {
              *(q-1)='\0';
              fprintf ( Fp, "%s%c%s", line, Force_Eol_Mark, MarginPtr );
              q=line; *(q++)=*(p-1); NbrChr=1; } }
   *q='\0'; return;
```

```
}
   void GetCgiEnv ()
{
   int i;
   for ( i=0; i<Nb_CGI_ENV_VAR; i++ ) *CgiEnv[i].VAL=getenv(CgiEnv[i].NAM);
   return;
}
   void PrtCgiEnv ( argc, argv )
   int argc; char *argv[];
{
   int i, j;
   if ( ! debug ) return; Html_Header();
   printf ( "<H2>%s / Argv, Env and Cgi parms</H2>\n", CGINAME);
   printf ("%s usage notes are available on the Net:\n", CGINAME);
   printf ("<A HREF=\"%s\" TARGET=\"_top\">\n", DocUrl);
   printf ("click here</A><BR><BR>\n");
/* Command line parameters */
   printf ( "<B><I>argc</I>=%d<P>\n<OL>\n",argc);
   for ( i=0; i<argc; i++ )
       {
         j = strlen(argv[i]);
         printf ( "<LI><I>argv[%d]</I>=%s (%d bytes)</LI>\n",i,argv[i],j ); }
/* UNIX Env variables */
   printf ( "</OL></B>\n<B><I>HTTP Env variables</B></I><P>\n<OL>\n");
   for ( i=0; i<Nb_CGI_ENV_VAR; i++ )
       {
         if ( strcmp (CgiEnv[i].NAM, "QUERY_STRING" ) == 0)
              printf ("<LI><B>QUERY_STRING= ** <I>See below</I> **</B></LI>\n");
         else printf ("<LI><B>%s</B>=%s</LI>\n",CgiEnv[i].NAM,*CgiEnv[i].VAL); }
   printf ( "</OL>\n"); return;
}
   void GetPrm (str, NbrPrm, FirstItm, MaxLen, P_Typ)
   unsigned char *str; int *NbrPrm, *MaxLen;
   struct CGI_PRM **FirstItm; char *P_Typ;
{
/* name & value should never be NULL ptr: empty strings should point to ""
   xlat tbl set to blank 0x00 to 0x1F & 0x80 to 0xa0, except 0x0a */
   struct CGI_PRM *Cgi_Prm; int lnam, lval, len;
   register unsigned char *p, *q; unsigned char sep, d1, d2, *name, *value;
   IniCgiPrm (&Cgi_Dflt_FROM,    Dir_FROM.name,    Dir_FROM.value);
   IniCgiPrm (&Cgi_Dflt_SUBJECT, Dir_SUBJECT.name, Dir_SUBJECT.value);
   if ( debug )
      {
        Html_Header(); len = strlen (str);
        printf  ("<B>CGI parameter block, type %s: %d byte(s)</B>\n",P_Typ,len);
        printf  ("</FONT><TT><BR><PRE>\n" ); stgdump ( str, len );
        printf  ("</PRE></TT><FONT FACE=\"%s\">\n", CGIFONT ); }
   *NbrPrm=0; if ( str == NULL || *str == '\0' ) return;
   Cgi_Prm_Prec=FirstItm; p=q=str; sep=0;
   while ( *p || sep )
     {
       sep = 0; name = q; lnam = 0; value = NULL; lval = 0;
       while ( 1 )
         {
           if      ( *p == '\0' )
                   { *(q++) = '\0'; break; }
           else if ( *p == '&' )
                   { *(q++) = '\0'; p++; sep=1; break; }
           else if ( *p == '=' && value == NULL)
                   { *(q++) = '\0'; value=q; p++; lnam=value-name-1; }
           else if ( *p == '+' )
                   { *(q++) = ' '; p++; }
           else if ( *p == '%' && isxdigit(d1=p[1]) && isxdigit(d2=p[2]) )
                   { *q = xlat [(hextbl[d1] << 4) + hextbl[d2]]; p+=3;
                     if ( *q == '\n' && value == NULL ) *q= ' '; q++; }
           else    { *(q++) = xlat [*(p++)]; } }
       if ( value == NULL ) { value=q-1; lnam = q-name-1; } else lval=q-value-1;
       *Cgi_Prm_Prec = Cgi_Prm=(struct CGI_PRM *)malloc(sizeof(struct CGI_PRM));
       Cgi_Prm->name  = name;      Cgi_Prm->lnam  = lnam;
       Cgi_Prm->value = value;     Cgi_Prm->lval  = lval;
       Cgi_Prm->prec  = D_NOSTR;   Cgi_Prm->state = 0;
       Cgi_Prm->next  = Cgi_Prm->nxtoc = Cgi_Prm->ptroc = END;
       Cgi_Prm->Refer = END_REF;   Cgi_Prm_Prec   = &Cgi_Prm->next;
       (*NbrPrm)++;
       if ( *MaxLen < lnam+lval ) *MaxLen = lnam+lval; }
   return;
}
   void  IniCgiPrm (Prm, name, value)
   struct CGI_PRM *Prm; char *name, *value;
{
/* Initialize CGI_PRM struct to default values */
   Prm->name  = name;     Prm->lnam  = strlen (name);
   Prm->value = value;    Prm->lval  = strlen (value);
   Prm->prec  = D_OMITT;  Prm->state = 0;
   Prm->next  = Prm->nxtoc = Prm->ptroc = END;
   Prm->Refer = END_REF;
   return;
}
   void PrtPrm ( NbrPrm, FirstItm, P_Typ )
   int NbrPrm; struct CGI_PRM *FirstItm; char *P_Typ;
{
   struct CGI_PRM *Cgi_Prm;
   if ( ! debug || NbrPrm==0 ) return; Html_Header();
   printf ( "<B>%d CGI Parms, type %s</B><P><OL>\n</FONT><TT>\n",NbrPrm,P_Typ );
   for ( Cgi_Prm=FirstItm; Cgi_Prm!=END; Cgi_Prm=Cgi_Prm->next )
     {
       printf ( "<LI>(%.4d,%.4d) <B>%s</B> = |%s|</LI>\n",
                Cgi_Prm->lnam, Cgi_Prm->lval, Cgi_Prm->name, Cgi_Prm->value ); }
   printf  ("</TT><FONT FACE=\"%s\">\n</OL>\n", CGIFONT ); return;
}
   Prt_Web_Dir ()
{
   int i; struct CGI_PRM *Cgi_Prm; struct MAILTO_PRM **Dir;
   if ( ! debug  || FirstPrm == END ) return; Html_Header();
   printf ( "<BR><B>Non recurrent instructions:</B><BR><BR>\n" );
   for ( Dir=Mailto_Prm, i=0; i<Nb_Mailto_Prm; i++, Dir++ )
      if ( (*Dir)->state & DIR_UNIQ )
         {
           if ( (*Dir)->code == FROM ) Prt_Chain ( &Cgi_Dflt_FROM );
           else printf("<B>%s =</B> \"%s\"<BR>\n",(*Dir)->name,(*Dir)->value); }
   if ( debug != 2 ) { printf ("<BR>\n"); return; }
   printf ("<BR><B>Mailing instruction(s) kept:</B><BR><BR>\n");
   Prt_Chain ( Dir_SUBJECT.first );  Prt_Chain ( Dir_TO.first );
   Prt_Chain ( Dir_TO_1X1.first );   Prt_Chain ( Dir_CC.first );
   Prt_Chain ( Dir_BCC.first );      Prt_Chain ( Dir_REPLY_TO.first );
   return;
}
   Prt_Chain (Prm)
   struct CGI_PRM *Prm;
{
   while  ( Prm != END )
          {
            printf ( "<B>%s = </B>", Prm->name);
            if ( *(Prm->prec) ) printf ( "%s = ", Prm->prec);
            printf ( "%s<BR>\n", Prm->value ); Prm=Prm->nxtoc; }
   return;
}
   Set_Maj (str)
   char *str;
{
   for (; *str; str++ ) *str = To_Upper [*str]; return;
}
   Set_Min (str)
   char *str;
{
   for (; *str; str++ ) *str = To_Lower [*str]; return;
}
   int Cnv_Edit (str)
   char *str;
{
   char *p;
   for ( p=str; *p=*(str++); p++ )
       {
         if ( *p == '\\' )
            {
              if      ( *str=='n' ) { *p='\n'; str++; }
              else if ( *str=='t' ) { *p='\t'; str++; } } }
   return;
}
   int Is_Blank (str)
   char *str;
{
   char c;
   while (*str) { if ( ( c=*(str++)) != '\n' && c != ' ' ) return 0; } return 1;
}
   int Val_Num (name, value)
   char *name, *value;
{
   int Num, nb, sign, decimal; char c, *p, *q;
   Num=1; nb=sign=decimal=0; p=q=value;
   while ( c = *(p++) = *(q++) )
     {
       if      ( isdigit (c) ) nb++;
       else if ( c == ' ' || c == '\n' ) p--;
       else if ( sign    == 0 && (c == '-' || c == '+') ) sign    = nb+1;
       else if ( decimal == 0 && (c == ',' || c == '.') ) decimal = 1;
       else      Num=0; }
   if ( sign > 1 && sign != nb+1 ) Num = 0;
   if ( Num == 0 )
      {
        sprintf ( WK, "This field should be numerical: \"%.*s=%.*s\"",
                  MAXVARL, name, MAXVARL, value );
        Err (WK, NULL,-1); }
   return Num;
}
   int No_Op (Cgi_Prm_Dir, Directive)
   struct CGI_PRM *Cgi_Prm_Dir; struct MAILTO_PRM *Directive;
{
   return;
}
   int ValIntPrm (Cgi_Prm_Dir, Directive)
   struct CGI_PRM *Cgi_Prm_Dir; struct MAILTO_PRM *Directive;
{
   int  cum, Valid; char *name, *value, c, *old, *new;
   name = Cgi_Prm_Dir->name; value = Cgi_Prm_Dir->value;
   cum=0; Valid=1; old=new=value;
   while ( c = *(new++) = *(old++) )
     {
       if      ( isdigit (c) && cum < MAXLINE ) cum = 10*cum + c - '0';
       else if ( c ==' ' ) new--;
       else      Valid=0; }
   if      ( *value == '\0' ) ;
   else if ( Valid == 0 || cum > MAXLINE )
      {
        sprintf(WK,"%s argument is not a positive integer =< %d: \"%.*s=%.*s\"",
                name, MAXLINE, MAXVARL, name, MAXVARL, value);
        Err (WK,name,0); Valid=0; }
   else if ( cum == 0 && Directive->code == LINE )
      {
        sprintf (WK,"%s argument cannot be 0",name); Err (WK,name,0); Valid=0; }
   else if ( (Cgi_Prm_Dir->state & PRM_IGNO) == 0 )
       {
         Directive->value = value; Directive->state |= DIR_EXIS; }
   return Valid;
```

```
}
   int Sav_As_Is (Cgi_Prm_Dir, Directive)
   struct CGI_PRM *Cgi_Prm_Dir; struct MAILTO_PRM *Directive;
{
   if ((Cgi_Prm_Dir->state & PRM_IGNO)==0) Directive->value=Cgi_Prm_Dir->value;
   return 1;
}
   int ValXfrUrl (Cgi_Prm_Dir, Directive)
   struct CGI_PRM *Cgi_Prm_Dir; struct MAILTO_PRM *Directive;
{
   char *p;
   if ((Cgi_Prm_Dir->state & PRM_IGNO)==0) Directive->value=Cgi_Prm_Dir->value;
   for ( p=Cgi_Prm_Dir->value; *p; p++ ) if ( *p==' ' || *p=='\n' ) *p = '+';
   return 1;
}
   int Val_E_Mail (Cgi_Prm_Dir, Directive)
   struct CGI_PRM *Cgi_Prm_Dir; struct MAILTO_PRM *Directive;
{
   int  i, fnd, Valid, Is_FROM;
   char *name, *value, *p, *r, *s, *ref, c; struct CGI_PRM *Prm;
   Is_FROM = (Directive->code == FROM);
   name    = Cgi_Prm_Dir->name;
   value   = Cgi_Prm_Dir->value;
   p=value; while ( *p==' ' ) p++; sprintf ( Prm_Buf, "\"%s=%s\"",name,value );
/* Arg is either an email addr or a reference to a field bearing such an addr */
   if ( *p != OPENREF )
      {
        Val_EM_Addr (value);
        if ( Is_FROM && (Cgi_Prm_Dir->state & PRM_IGNO) == 0 )
           {
             Cgi_Dflt_FROM.prec =              (*value) ? D_NOSTR : D_OMITT;
             Cgi_Dflt_FROM.value=Dir_FROM.value=(*value) ? value   : D_EMAIL; }
        return; }
/* Chain _CC _BCC _REPLY_TO with reference {_FROM}. Use prtoc idle on these */
   sprintf ( WK, "{%s}", ref=Dir_FROM.name );
   if ( strncmp ( p, WK, i=strlen(WK) ) == 0 )
      {
        if ( (Directive->state & DIR_FROM) == 0 )
           {
             sprintf ( WK,"Reference {%s} is invalid on %s: %.*s",
                       ref,name,2*MAXVARL,Prm_Buf );
             Err (WK, name,0); return; }
        p+=i; while ( *p == ' ' ) p++;
        if ( *p )
           {
             sprintf ( WK,"No text allowed after {%s} in %.*s",
                       ref,2*MAXVARL,Prm_Buf );
             Err (WK, name,0); return; }
        if    (Dir_FROM.first == END) Dir_FROM.first = Cgi_Prm_Dir;
        else  (Dir_FROM.last)->ptroc = Cgi_Prm_Dir;
        Dir_FROM.last = Cgi_Prm_Dir; Cgi_Prm_Dir->prec = value; return; }
```

```
/* Track & process ref other than {_FROM}, checking first/last non blank char */
   r=p; s=p+1; i=0; while(*(++r)) i++; while(i && *(--r)==' ') i--; *(s+i)='\0';
   if ( *r != CLOSREF || *s == '_' )
      {
        sprintf ( WK, "Invalid reference: %.*s", 2*MAXVARL, Prm_Buf );
        Err (WK, NULL,0); return; }
   for ( *r='\0', fnd=0, Valid=1, Prm=FirstPrm; Prm!=END; Prm=Prm->next )
       {
         if ( strcmp(s,Prm->name) == 0 )
            {
              fnd=1; sprintf ( Prm_Buf, "\"%s=%s\"", Prm->name, Prm->value );
              if ( (Valid=Val_EM_Addr(Prm->value)) == 0 ) break;
              if ( *(Prm->value) == '\0' ) continue;
              if   ( Is_FROM && (Cgi_Prm_Dir->state & PRM_IGNO) == 0 )
                   {
                     Cgi_Dflt_FROM.prec  = value;
                     Cgi_Dflt_FROM.value = Dir_FROM.value = Prm->value; }
              else { Cgi_Prm_Dir->prec   = value;
                     Cgi_Prm_Dir->value  = Prm->value; }
              *r=CLOSREF; return; } }
   *r = CLOSREF;
   if ( fnd == 0 )
      {
        sprintf ( WK, "Reference to an unknown field: %.*s",2*MAXVARL,Prm_Buf );
        Err (WK, NULL,0); }
   else if ( Valid == 0 )
      {
        sprintf ( WK, "Reference to an invalid address: \"%.*s=%.*s\"",
                  MAXVARL, name, MAXVARL, value );
        Err (WK, NULL,0); }
   *value = '\0'; return;
}
   int Val_EM_Addr (value)
   char *value;
{
/* Email address validation using addr-spec rules from RFC-822
   addr-spec  : local-part "@" domain
   local-part : word *("." word)
   domain     : atom *("." atom)
   word       : atom | string
   string     : <"> *( <\>char | char_excluant " \ CR ) <">
   atom       : 1*<char_excluant Dec(0-31,127) [ ] ( ) < > : ; blanc , . @ " \ >
   char       : 7-bit ascii char
   Mail_Chr_Tbl code chart
   0x00 = End-of-email-address marker        0x2e = ASCII dot char   <.>
   0x01 = ASCII chars valid everywhere       0x40 = ASCII AT char    <@>
   0x02 = ASCII chars invalid everywhere     0x22 = ASCII quote char <">
   0x20 = ASCII blank char < > */
   int LocDom; char code, c, *p, *q, *start;
   start=value; while ( *start == ' ' ) start++;
   p=start; q=value; LocDom=0;
   while ( code = Mail_Chr_Tbl[ c = *(q++) = *(p++) ] )
     {
       if ( code == 1 )   continue;
       if ( code == 2 ) { EM_Err ( c, "invalid" ); return 0; }
       if ( code == '.' )
          {
            if ( p-start<=1 || *p=='\0' || *p=='@' )
               { EM_Err ( c, "misplaced" ); return 0; }
            start = p; continue; }
       if ( code == '@' )
          {
            if ( LocDom==1 || p-start<=1 || *p=='\0' )
               { EM_Err ( c, "misplaced" ); return 0; }
            LocDom=1; start=p; continue; }
       if ( code == ' ' )
          {
            while ( *p == ' ' ) p++;
            if ( *p ) { EM_Err ( c, "invalid" ); return 0; }
            *(q-1) = '\0'; break; }
       if ( code == '"' )
          {
            if ( LocDom == 1 ) { EM_Err ( c, "misplaced" ); return 0; }
            if ( *p == '"' ) { q--; p++; continue; }
            while ( (c=*(q++)=*(p++)) != '"' )
              {
                if ( c=='\\' && *p ) *(q++)=*(p++);
                if ( c=='\0') { EM_Err ( '"', "not matched" ); return 0; } } } }
   return 1;
}
   EM_Err (chr, msg)
   char chr, *msg;
{
   sprintf ( WK, "Character «%c» is %s in email address: %.*s",
             chr, msg, 2*MAXVARL, Prm_Buf );
   Err (WK, "EMAIL",0);
   return;
}
   int Val_IF_Struct (Prm_IF, Directive)
   struct CGI_PRM *Prm_IF; struct MAILTO_PRM *Directive;
{
   int Res_Glob, Res_Sect, Res_Last, Else_Fnd, code, flags, state; char *name;
   struct CGI_PRM *Prm; struct MAILTO_PRM *Dir;
/* Stmnts _IF/_ELSE_IF may be followed by any number of _AND/_OR in any order.
   _AND has precedence over _OR. All cond proposals are syntax checked even if
   final result is already known. Bypassed statements are also syntax checked */
   Prm = Prm_IF; Res_Glob = Else_Fnd = 0;
   while ( Prm != END )
     {
       Prm->state |= PRM_INVI | PRM_SYST;
       if ( Dir->code == END_IF  ) break;
       if ( Else_Fnd ) BLIF_Err (Prm, "Statement unexpected after _ELSE");
```

```
      if   ( Dir->code == ELSE ) { Res_Sect=Else_Fnd=1; Prm=Prm->next; }
      else { Res_Sect = Res_Last = Val_Cond (Prm);
             for ( Prm=Prm->next; Prm != END; Prm=Prm->next )
                 {
                   if      ( strcmp (Dir_AND.name, Prm->name) == 0 ) code=AND;
                   else if ( strcmp (Dir_OR.name,  Prm->name) == 0 ) code=OR;
                   else break;
                   Prm->state |= PRM_INVI | PRM_SYST;
                   if      ( Res_Last < 0 ) Val_Cond (Prm);
                   else if ( (Res_Last=Val_Cond (Prm)) < 0 ) Res_Sect=Res_Last;
                   else if ( code == AND ) Res_Sect &= Res_Last;
                   else if ( Res_Sect ) Res_Last = -1;
                   else      Res_Sect = Res_Last; } }
      if ( Res_Glob==0 && Res_Sect==1 ) flags=0; else flags=PRM_IGNO|PRM_INVI;
      if ( Res_Sect==1 ) Res_Glob=1;
      for ( ; Prm != END; Prm=Prm->next )
          {
            if ( (Dir=Get_Dir_Ptr (Prm)) == END_DIR ) continue;
            if ( (state=Dir->state) & DIR_LOGI )
               {
                 BLIF_Err (Prm, "Misplaced statement in current IF-block");
                 Prm->state |= PRM_INVI | PRM_SYST; continue; }
            if ( state & DIR_SECT ) break;
            Prm->state |= flags; } }
   if ( Prm == END ) BLIF_Err ( Prm_IF, "Closing _END_IF missing for" );
   return;
}
   int Val_Cond (Cond_Prm)
   struct CGI_PRM *Cond_Prm;
{
   int i, fnd, cond; char *r, *s, *p, *q, *name, *value; struct CGI_PRM *Prm;
   name = Cond_Prm->name; value = Cond_Prm->value;
/* Validate name, operator, value & test all instances of name against value
   Return 1 if all instances pass test, 0 if at least 1 fails, -1 if error */
   sprintf ( Prm_Buf, "\"%s=%s\"", name, value );
   sprintf ( WK, "Invalid reference {...} on: %.*s", 2*MAXVARL, Prm_Buf );
   p=value; while ( *p == ' ' ) p++;
   if ( *p != OPENREF ) { Err (WK, "_IF",0); return -1; }
   r=p; s=p+1; while (*(++r) && *r != CLOSREF);
   if ( *r != CLOSREF || *s == '_' ) { Err (WK, "_IF",0); return -1; }
   for ( cond=0, q=r+1, i=0; i<Nb_Opr; i++ )
       {
         if (strncmp(q,Log_Opr[i].name,4)==0) { cond=Log_Opr[i].cond; break; } }
   if ( cond == 0 )
      {
        sprintf ( WK, "Invalid operator on: %.*s", 2*MAXVARL, Prm_Buf );
        Err (WK, "_IF",0); return -1; }
   for ( fnd=0, *r='\0', q+=4, Prm=FirstPrm; Prm!=END; Prm=Prm->next )
       {
         if ( strcmp(s, Prm->name) == 0 )
            {
              fnd=1; i=strcmp (Prm->value, q);
              if      ( i == 0 ) i = Is_EQ;
              else if ( i <  0 ) i = Is_LT;
              else               i = Is_GT;
              if ( i & cond ) continue; else { *r = CLOSREF; return 0; } } }
   *r = CLOSREF;
   if ( fnd == 0 )
      {
        sprintf ( WK,"Reference to an unknown field: %.*s",2*MAXVARL,Prm_Buf );
        Err (WK, "_IF",0);  return -1; }
   return 1;
}
   struct MAILTO_PRM *Get_Dir_Ptr (Prm)
   struct CGI_PRM *Prm;
{
   int i, j; char *name;
   name = Prm->name;
   if ( *name != '_' )
      {
        BLIF_Err ( Prm, "User field unauthorized between _IF and _END_IF" );
        return END_DIR; }
   for ( j=1, i=0; j>0 && i<Nb_Mailto_Prm; i++ )
       {
         if ( (j=strcmp(name,Mailto_Prm[i]->name))==0 ) return Mailto_Prm[i]; }
   BLIF_Err ( Prm, "Invalid instruction" ); Prm->state |= PRM_INVI | PRM_SYST;
   return END_DIR;
}
   BLIF_Err (Prm, msg)
   struct CGI_PRM *Prm; char *msg;
{
   sprintf ( WK,"%s: \"%.*s=%.*s\"",msg,MAXVARL,Prm->name,MAXVARL,Prm->value );
   Err (WK, "_IF" , 0);
   return;
}
   int Val_Refs (Cgi_Prm_Dir, Directive)
   struct CGI_PRM *Cgi_Prm_Dir; struct MAILTO_PRM *Directive;
{
/* Track/validate/chain references initiated by OPENREF & terminated by CLOSREF
   Literal OPENREF, CLOSREF & ESC_REF should be preceded by ESC_REF
   When a field is referenced, chain its successive occurrences
   Mask any _WRITE_COND bearing a reference to an unfilled field */

   int    fnd, AllEmpty, One_Missing, Is_FROM, Is_SysRef;
   char   *name, *value, c, *p, *q, *start, *stop, *Org_Prm;
   struct CGI_PRM *Cgi_Prm, *Cgi_Head_Prm, *Cgi_Top_Prm, **Cgi_Last_Prm;
   struct REF_PRM *Ref;
   name = Cgi_Prm_Dir->name; value = Cgi_Prm_Dir->value;
   sprintf (Prm_Buf, "Invalid reference: \"{\" or \"}\" misplaced in ");
   Org_Prm = Prm_Buf + strlen (Prm_Buf);
   sprintf (Org_Prm, "\"%s=%s\"", name, value);
   sprintf (WK, "Invalid reference {...} on %.*s", 2*MAXVARL, Org_Prm);
```

```
p=q=value; start=stop=NULL; One_Missing=0;
while ( c = *(q++) = *(p++) )
  {
    if      ( c==ESC_REF ) { *(q-1)=*p; if ( *(p++) ) continue; else break; }
    else if ( c==CLOSREF ) { Err (Prm_Buf, "CHAP5",0); return; }
    else if ( c!=OPENREF ) { continue;  }
    else    { start = q; }
    while ( c = *(q++) = *(p++) )
      {
        if      ( c == ESC_REF )
                { *(q-1) = *p; if ( *(p++) ) continue; else break; }
        else if ( c == OPENREF ) { Err (Prm_Buf, "CHAP5",0); return; }
        else if ( c != CLOSREF ) { continue;  }
        else    { stop = q-1; *stop='\0'; break; } }
    if ( stop == NULL ) { Err (Prm_Buf, "CHAP5",0); return; }
    Is_SysRef=Is_FROM=fnd=0; AllEmpty=1; Cgi_Prm=FirstPrm;
    if ( *start == '_' )
       {
         if ( *(start+1) == '$' )
            {
              if   ( (Cgi_Prm=Sys_Refs (start)) != END ) Is_SysRef=1;
              else { Err (WK, "CHAP5",0); return; } }
         else if ( strcmp (Dir_FROM.name, start) )
                { Err (WK, "CHAP5",0); return; }
         else Is_FROM=1; }
    for ( ; Cgi_Prm!=END; Cgi_Prm=Cgi_Prm->next )
       {
         if ( strcmp(start,Cgi_Prm->name) == 0 )
            {
              if ( fnd == 0 )
                 {
                   fnd = 1; Cgi_Head_Prm = (Cgi_Top_Prm=Cgi_Prm)->ptroc;
                   if ( Cgi_Top_Prm->state & PRM_NULL ) break;
                   if ( Cgi_Head_Prm != END_REF ) { AllEmpty=0; break; } }
              if ( Is_FROM && Is_Blank(Cgi_Prm->value) )
                 { AllEmpty=1; continue; }
              if ( *(Cgi_Prm->value) == '\0' ) continue;
              if ( AllEmpty )
                 {
                   Cgi_Top_Prm->ptroc = Cgi_Prm;
                   Cgi_Last_Prm = &Cgi_Head_Prm; AllEmpty = 0; }
              *Cgi_Last_Prm = Cgi_Prm; Cgi_Last_Prm  = &Cgi_Prm->nxtoc;
              if ( Is_SysRef) break; } }
    if ( Is_FROM )
       {
         Cgi_Head_Prm=&Cgi_Dflt_FROM;
         if ( fnd && AllEmpty ) Cgi_Top_Prm->state |= PRM_NULL;
         if ( (fnd == 0 || AllEmpty) && Directive->code != WRITE_COND )
            { fnd=1; AllEmpty=0; } }
    if ( fnd == 0 )
       { *stop=CLOSREF; start=stop=NULL; One_Missing=1; continue; }
```

```
        if ( AllEmpty )
           {
             Cgi_Top_Prm->state |= PRM_NULL; q=start-1;
             start=stop=NULL; One_Missing=1; continue; }
        NbrRef++;
        *Ref_Last  = Ref = (struct REF_PRM *) malloc (sizeof(struct REF_PRM));
        Ref->targ  = Cgi_Head_Prm;
        Ref->next  = END_REF;
        Ref_Last   = &Ref->next;
        if ( Cgi_Prm_Dir->Refer == END_REF ) Cgi_Prm_Dir->Refer = Ref;
        *(start-1) = PTR_REF; q=start; start=stop=NULL; }
    if ( *value=='\0' || (Directive->code == WRITE_COND && One_Missing) )
         Cgi_Prm_Dir->state |= PRM_INVI;
    return;
}
    struct CGI_PRM *Sys_Refs (name)
    char *name;
{
    int i, j; struct CGI_PRM *Prm;
    for ( i=0; i<Nb_Sys_Var && (j=strcmp(name,Sys_Var[i].name))>0; i++ );
    if ( j ) return END;
    if ( (Prm=Sys_Var[i].PrmLoc) != END ) return Prm;
    if ( *(Sys_Var[i].value) == '\0' ) (Sys_Var[i].SetVal) (i);
    Sys_Var[i].PrmLoc = Prm = (struct CGI_PRM *) malloc(sizeof(struct CGI_PRM));
    IniCgiPrm (Prm, Sys_Var[i].name, Sys_Var[i].value);
    Prm->state |= PRM_INVI | PRM_SYST;
    NbrWrkPrm++; *Cgi_Prm_Prec = Prm; Cgi_Prm_Prec = &Prm->next;
    return Prm;
}
    Sys_Dates (NoVar)
    int NoVar;
{
    struct tm *timeptr; time_t timer;
    timer = time((time_t *)NULL); timeptr = localtime(&timer); timeptr->tm_mon++;
    sprintf  ( _$YRDAY,    "%3.3d",    timeptr->tm_yday);
    sprintf  ( _$WKDAY,    "%s",       WkDays[timeptr->tm_wday] );
    sprintf  ( _$MONTH,    "%s",       ymonth[timeptr->tm_mon-1] );
    strftime ( _$YEAR, 5,  "%Y",       timeptr);
    strftime ( _$YY,   3,  "%y",       timeptr);
    sprintf  ( _$MM,       "%2.2d",    timeptr->tm_mon );
    sprintf  ( _$DD,       "%2.2d",    timeptr->tm_mday );
    sprintf  ( _$HRS,      "%2.2d",    timeptr->tm_hour );
    sprintf  ( _$MIN,      "%2.2d",    timeptr->tm_min );
    sprintf  ( _$SEC,      "%2.2d",    timeptr->tm_sec );
    sprintf  ( _$DATE,     "%s/%s/%s", _$YEAR,  _$MM,  _$DD );
    sprintf  ( _$TIME,     "%s:%s:%s", _$HRS,   _$MIN, _$SEC );
    return;
}
    Sys_Refer (NoVar)
    int NoVar;
{
   Sys_Var[NoVar].value = Cgi_HTTP_REFERER; return;
}
   Sys_Noseq (NoVar)
   int NoVar;
{
   int fdi, fdo, j; long noseq; char *rec, buf[MAXLNSQ];
/* To open NSQFILE, use O_NSHARE locking feature to serialize concurrent uses */
   sprintf ( _$NOSEQ, "{_$NOSEQ}" );
   for ( j=0; (fdi=open(NSQFILE, O_RDWR|O_NSHARE,0700))<0; sleep (TIMWAIT) )
       { if ( ++j < TIMNOPN ) continue;
         Log ("Cannot open file fd = NSQFILE"); Log (syserr); return; }
   if ( (fdo = open (TMPFILE, O_CREAT|O_RDWR|O_TRUNC, 0700) ) < 0 )
      { Log ("Cannot open file fd = TMPFILE"); close (fdi) ; return; }
   Noseq_File.Fd    = fdi;
   Noseq_File.Buf   = Noseq_File.Pos = Noseq_Fd_Buf = (char *) malloc (FDBUFLN);
   Noseq_File.Balan = 0;
   for ( j=1; (rec=fdgets (&Noseq_File)) != NULL; )
       { if ( (j=strcmp(PagNam, rec+MAXDGIT+1)) > 0 )
            { *(Noseq_File.Pos-1)='\n'; write ( fdo, rec, Noseq_File.LasRl ); }
         else break; }
   if ( j==0 ) noseq=Read_Noseq(rec)+1; else noseq=1;
   sprintf ( buf, "%*.*ld,%s\n", MAXDGIT, MAXDGIT, noseq, PagNam );
   write   ( fdo, buf, strlen(buf) );
   if ( j<0 ) { *(Noseq_File.Pos-1)='\n'; write (fdo, rec, Noseq_File.LasRl ); }
   while ( (rec=fdgets (&Noseq_File)) != NULL )
         { *(Noseq_File.Pos-1)='\n'; write (fdo, rec, Noseq_File.LasRl ); }
   sprintf ( _$NOSEQ, "%ld", noseq );
   if ( debug == 0 )
      {
       lseek (fdi,OL,SEEK_SET); lseek (fdo,OL,SEEK_SET);
       while ( (j = read (fdo, Noseq_Fd_Buf, FDBUFLN)) != 0 )
             { if ( j < 0 || write (fdi, Noseq_Fd_Buf, j) != j )
                  { Log ("Problem copying TMPFILE onto NSQFILE"); break; } } }
   close (fdi); close (fdo); return;
}
   char *fdgets (Fs)
   struct FD_REC *Fs;
{
   int Balan, Bsiz, Nchr; char *Buf, *Rec, *Pos;
   Balan = Fs->Balan; Bsiz = Fs->Bsiz; Buf = Fs->Buf; Pos = Rec = Fs->Pos;
   while (1)
     {
       while ( Balan-- )
         {
           if ( *(Pos++) == '\n' )
              {
                Fs->Pos   = Pos;      Fs->Balan = Balan;
                Fs->LasRl = Pos-Rec;  *(Pos-1)  = '\0';   return Rec; } }
       if ( (Nchr=Pos-Rec) >= Bsiz ) Err ( "NSQFILE: Record too long", NULL,1);
       memcpy (Buf, Rec, Nchr);
       Rec = Buf; Pos = Buf + Nchr; Balan = read (Fs->Fd, Pos, Bsiz-Nchr);
```

```
        if      ( Balan == 0 ) return NULL;
        else if ( Balan <  0 ) Err ( "Error reading NSQFILE", NULL,1);
        else      continue; }
}
   long Read_Noseq (str)
   char *str;
{
   long cum;
   for ( cum=0; isdigit (*str); str++ ) cum = 10*cum + *str - '0'; return cum;
}
   Delete_Dup (Chain_Head)
   struct CGI_PRM *Chain_Head;
{
   struct CGI_PRM *Prm, *Prma, *Prmb;
/* Delete duplicates within chain starting at Chain_Head */
   for ( Prma=Chain_Head; Prma != END; Prma=Prma->nxtoc )
       {
         for ( Prm=Prma, Prmb=Prma->nxtoc; Prmb != END; Prmb=Prmb->nxtoc )
             {
               if ( strcmp(Prma->value,Prmb->value)==0 ) Prm->nxtoc=Prmb->nxtoc;
               else Prm=Prmb; } }
   return;
}
   Verif_Dir_Subject (Directive)
   struct MAILTO_PRM *Directive;
{
   struct CGI_PRM *Prm, *Prm_a; struct REF_PRM *Ref; char *str;
   for ( Prm=Directive->first; Prm != END; Prm=Prm->nxtoc )
       {
         for ( Ref=Prm->Refer, str=Prm->value; *str; str++ )
             {
               if ( *str == PTR_REF && Ref != END_REF )
                  {
                    if ( !Is_Blank ((Prm_a=Ref->targ)->value) ) return;
                    while ( (Prm_a=Prm_a->nxtoc) != END )
                            if ( !Is_Blank (Prm_a->value) ) return;
                    Ref = Ref->next; }
               else if ( *str != '\n' && *str != ' ' ) return; } }
   Dir_SUBJECT.first = &Cgi_Dflt_SUBJECT; return;
}
   Chk_HTTP_REFERER ()
{
   int i, len, naml, fnd; char *p, *q, *s;
   *PagNam = '\0'; fnd=0;
   if ( Cgi_HTTP_REFERER != NULL ) p=Cgi_HTTP_REFERER; else p=PagNam;
   for ( q=PagNam, i=0; i<8 && *p!='\0'; i++ ) *(q++)=To_Lower[*(p++)]; *q='\0';
   if ( strncmp (PagNam, "http://",7) == 0
   ||   strcmp  (PagNam, "https://")  == 0 )
      {
        p=Cgi_HTTP_REFERER + ((PagNam[4]=='s') ? 8 : 7); q=PagNam; naml=0;
        while ( *p != '/' && *p !=':' && *p !='\0' )
```

```
            {
              *(q++)=To_Lower [*(p++)];
              if ( ++naml>MAXDOML )
                   Err ( "Referer domain name too long: MAXDOML", NULL,1); }
          for ( *q='\0', i=0; i<Nb_Dom; i++ )
              {
                if ( (len = naml - strlen (Dom_Name_Lst[i])) < 0 ) continue;
                for ( s=Dom_Name_Lst[i]; *s; s++ ) *s=To_Lower[*s];
                if ( strcmp(PagNam+len,Dom_Name_Lst[i]) == 0 ) { fnd=1; break; } }
          while ( *p != '/' && *p !='\0' ) p++;
          while ( *p != '?' && *p !='\0' )
            {
              *(q++)=*(p++); /* *(q++)=To_Lower [*(p++)]; */
              if ( ++naml>MAXREFE ) Err ( "Referer too long: MAXREFE", NULL,1); }
          *q='\0'; }
     if ( fnd || DOMCHCK == 0 || debug ) return 0;
     return 1;
  }
     int NoHelp ()
  {
     Html_Header();
     printf ("<CENTER><BR><BR><HR SIZE=5 WIDTH=95%%><BR>\n<FONT SIZE=+1>");
     printf ("<B>%s online usage notes not installed\n", CGINAME);
     printf ("<BR><BR>Please contact site webmaster\n");
     printf ("<BR><BR>Serveur name: %s<BR><BR>\n", Cgi_SERVER_NAME);
     printf ("</B></FONT>\n<HR SIZE=5 WIDTH=95%%><BR></CENTER>\n");
     fflush (stdout); Free_Mem(); exit (0);
  }
     void Redirect (New_Url)
     char *New_Url;
  {
     printf ("Location: %s\n\n",New_Url); fflush (stdout); Free_Mem(); exit (0);
  }
     void Html_Header()
  {
     if (Html_Header_Flag ) return; else Html_Header_Flag = 1;
     printf ( "content-type: text/html\n\n");
     printf ( "<HTML>\n<HEAD>\n<TITLE>%s</TITLE>\n</HEAD>\n", CGINAME);
     printf ( "<BODY BGCOLOR=WHITE>\n<FONT FACE=\"%s\">\n", CGIFONT );
     return;
  }
     void Html_Trailer()
  {
     if ( ! Html_Header_Flag ) return;
     if ( debug ) printf ( "<B>The end / %s</B><BR><BR>\n\n", CGINAME);
     printf ( "</BODY>\n</HTML>\n" ); fflush (stdout); return;
  }
     int Log (msg)
     char *msg;
  {
     FILE *fptr;
```

```
   if ( (fptr=fopen(LOGFILE, "a" )) == NULL ) return;
   fprintf ( fptr, "%s (%s) %s\n", datetime(datbuf), Cgi_REMOTE_ADDR, msg );
   fclose  ( fptr ); return;
}
   int Err (msg, aide, stop)
   char *msg, *aide; int stop;
{
/* stop<=0: Prt msg & continue processing
   stop >0: Prt msg for webmaster/1, form designer/2 or form filler/3 & exit */
   Html_Header();
   if ( Error_Flag == 0 )
      {
        printf ("<BR><HR SIZE=5 WIDTH=100%%><BR>\n<FONT SIZE=+1><B>\n");
        printf ("%s cannot process this form<BR>\n", CGINAME);
        printf ("for the following reason(s):\n</B></FONT>\n<UL>\n"); }
   if ( stop<0 && Error_Flag!=2 ) Error_Flag = 3; else Error_Flag = 2;
   if ( msg  != NULL ) printf ("<LI><B>%s</B>\n",msg);
   if ( aide != NULL )
      {
        printf ("<B> &nbsp;&nbsp;(<A HREF=\"%s#%s\" TARGET=\"_top\">",
                DocUrl, aide);
        printf ("Help</A>)</B>\n"); }
   if ( stop <= 0 ) return; else printf ("<BR><BR><FONT SIZE=+1>\n</UL>\n");
   if      ( stop == 1 )
      { printf ("Contact webmaster, specify page \"%s\"\n",Cgi_HTTP_REFERER ); }
   else if ( stop == 2 )
      { printf ("Contact designer of faulty page \"%s\"\n",Cgi_HTTP_REFERER ); }
   else if ( stop == 3 )
      { printf ("Make appropriate corrections and resubmit form\n" ); }
   printf ("</FONT>\n<BR><BR><FONT SIZE=+1>\n");
   printf ("%s usage notes are available on the Net:\n", CGINAME);
   printf ("<A HREF=\"%s\" TARGET=\"_top\">\n", DocUrl);
   printf ("click here</A><BR><BR>\n</FONT>\n<HR SIZE=5 WIDTH=100%%><BR>\n");
   if ( debug==2 )
      {
        PrtPrm ( Nbr_Prm,   FirstPrm,    CGINAME );
        PrtPrm ( NbrWrkPrm, FirstWrkPrm, WrkDsc ); }
   Prt_Web_Dir (); Free_Mem(); Html_Trailer(); exit (0);
}
   void Free_Mem ()
{
   struct CGI_PRM *Prm; struct REF_PRM *Ref;
   int sp=sizeof(struct CGI_PRM), sr=sizeof(struct REF_PRM); void *ptr;
   if ( debug ) { Html_Header();
                  printf ( "<BR><B>Free_Mem: %d block(s) of ", NbrUrlPrm );
                  printf ( "%d bytes, type %s</B><BR>\n", sp, UrlDsc ); }
   for ( Prm=FirstUrlPrm; Prm!=END; Prm=Prm->next, free(ptr) ) ptr=(void*) Prm;
   if ( debug ) { printf ( "<B>Free_Mem: %d block(s) of ", NbrFrmPrm );
                  printf ( "%d bytes, type %s</B><BR>\n", sp, FrmDsc ); }
   for ( Prm=FirstFrmPrm; Prm!=END; Prm=Prm->next, free(ptr) ) ptr=(void*) Prm;
   if ( debug ) { printf ( "<B>Free_Mem: %d block(s) of ", NbrWrkPrm );
                  printf ( "%d bytes, type %s</B><BR>\n", sp, WrkDsc ); }
   for ( Prm=FirstWrkPrm; Prm!=END; Prm=Prm->next, free(ptr) ) ptr=(void*) Prm;
   if ( debug ) { printf ( "<B>Free_Mem: %d block(s) of ", NbrRef );
                  printf ( "%d bytes, type REF</B><BR>\n", sr ); }
   for ( Ref=Ref_First; Ref!=END; Ref=Ref->next, free(ptr) ) ptr=(void*) Ref;
   if ( Cgi_Stdin_Stream != NULL )
      { free ( (void *) Cgi_Stdin_Stream );
        if ( debug ) printf (freefmt,CgiCntLen,"STDIN" ); }
   if ( Prm_Buf != NULL )
      { free ( (void *) Prm_Buf );
        if ( debug ) printf (freefmt,Prm_Buf_Len,"Prm_Buf" ); }
   if ( Noseq_Fd_Buf != NULL )
      { free ( (void *) Noseq_Fd_Buf );
        if ( debug ) printf (freefmt,FDBUFLN,"Noseq_Fd_Buf" ); }
   if ( debug ) printf ( "<BR>\n" );
   return;
}
   char *datetime (date)
   char *date;
{
   struct tm *timeptr; time_t timer; char year[32];
   timer = time((time_t *)NULL); timeptr = localtime(&timer); timeptr->tm_mon++;
   strftime (year,5,"%Y",timeptr);
   sprintf ( date, "%3.3s %.2d/%.2d/%.2d %.2d:%.2d:%.2d",
                    WkDays[timeptr->tm_wday],
                    timeptr->tm_year + 1900, timeptr->tm_mon, timeptr->tm_mday,
                    timeptr->tm_hour, timeptr->tm_min, timeptr->tm_sec );
   return date;
}
   void stgdump (adr,len)
   int len, * adr;
{
   int i, j, nc=4*sizeof(int); char *chr=(char *)adr, visual[4*sizeof(int)+1];
/* Print on stdout storage dump starting at adr for len consecutive bytes */
   Html_Header(); visual[nc] = '\0';
   for (i=0; i<len; i+=nc, adr+=4, chr+=nc)
       {
         for (j=0; j<nc; j++) visual[j]=(chr[j]>31&&chr[j]<127) ? chr[j] : '.';
         printf ("%08x : %08x %08x %08x %08x | %s | %08x\n",
                 adr,*adr,*(adr+1),*(adr+2),*(adr+3),visual,adr); }
   return;
}
```

Pierre Croisetiere
System Analyst (Canada) © Xephon 2000

AIX news

IBM has announced Communications Server for AIX Version 6.0, which now supports SSL with client authentication, SLP for service location and load balancing for TN Servers, and TN Redirector for host access to TN clients. SNA/networking extensions include Enterprise Extender, Branch Extender, and MPC+ channel connectivity. There's also a Host Publisher feature, which provides Web-to-host facilities. New APIs include 64-bit support and CPI-C for Java applications. Out now, it's priced at US$1,000.

The company also announced HACMP Version 4.4.0. The product has improved monitoring of the state of applications and can (if necessary) restart an application or fail an application group to another node. Tivoli Cluster Monitoring monitors the state of an HACMP cluster and its components on a Tivoli Desktop window.

It comes with an enhanced LVM task guide, which provides a display of the physical location of each available disk, and an NFS tool for migrating between the HACMP 4.3.1's HANFS and HACMP 4.4's HAS.

It's available now on AIX 4.3.3 or higher on RS/6000s with at least four slots. RS/6000 uniprocessors and SMP servers are supported, as are SP systems (a new release of PSSP that supports HACMP 4.4 is to be released in 2H00). Prices start at $4,500.

For further information, contact your local IBM representative.

* * *

MQSoftware has released QPasa! Version 2.2.1, a tool that provides analysis and management of the configuration, performance, and operation of MQSeries, MQSI Versions 1 and 2, MQSeries Workflow, and MQSeries Everyplace. The Message Manager allows users to browse and edit messages on queues, including dead-letter and alias queues, while application developers can use Message Manager to simulate messages during development. The Extensible Agent API and SDK allows users to create their own extensions to monitor data not handled by QPasa!

In addition to AIX, the product also supports MQSeries installations on HP-UX, Linux, Solaris, Windows NT, and Windows 2000. It's out now, and prices are available on request from the vendor.

For further information contact:
MQSoftware Inc, 7575 Golden Valley Road, Suite 140, Minneapolis, MN 55427, USA
Tel: +1 612 546 9080
Fax: +1 612 546 9082
Web: http://www.mqsoftware.com

MQSoftware Europe Ltd, The Surrey Technology Centre, 40 Occam Road, Surrey Research Park, Guildford, Surrey GU2 5YH, UK
Tel: +44 1483 295400
Fax: +44 1483 573704

* * *

xephon