

STARS

The *BOEING* Company
D613-10410

CDRL 000410

2

AD-A240 476



**Software Technology for Adaptable,
Reliable Systems (STARS)**

Submitted to:
Electronic Systems Division
Air Force Systems Command, USAF
Hanscom AFB, MA 01731-5000

Contract No:
F-19628-88-D-0028

**CDRL 000410
Enhanced Prototype Capability
(Ada Source Code)**

March 7, 1989

The *BOEING* Company
Boeing Military Airplanes Division
P.O. Box 7730
Wichita, Kansas 67277-7730

Approved for public release - distribution is unlimited

Enhanced Prototype Capability
(Ada Source Code)
D613-10410

91-10152



REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 07-MAR-89		3. REPORT TYPE AND DATES COVERED
4. TITLE AND SUBTITLE Enhanced Prototype Capability (Ada Source Code)			5. FUNDING NUMBERS C: F19628-88-D-0028	
6. AUTHOR(S) Charles Elsner Kent Brummer			TA: BQ-12	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) The Boeing Company Boeing Aerospace and Electronics Division Systems and Software Engineering P.O. Box 3999 Seattle, Washington 98124			8. PERFORMING ORGANIZATION REPORT NUMBER D-613- 10410	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) ESD/AVS Bldg. 17-04 Room 113 Hanscom Air Force Base, 01731-5000			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release - distribution unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (Maximum 200 words)				
14. SUBJECT TERMS Keywords: STARS Enhanced Prototype Capability			15. NUMBER OF PAGES 64	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT None	

Enhanced Prototype Capability
(Ada Source Code)

Prepared by:

Charles Elsner 6-March-89
Charles Elsner
Repository Software Development

Prepared by:

Kent Brummer 3/7/89
Kent Brummer
Repository Chief Programmer

Reviewed by:

for John Neorr 3/8/89
M. J. Davis
Q-Task 12 Manager

Reviewed by:

James E. King
James E. King
System Architect

Reviewed by:

John M. Neorr 3/8/89
John M. Neorr
Development Manager

Approved by:

William M. Hodges 3/8/89
William M. Hodges
STARS Program Manager



Enhanced Prototype Capability
(Ada Source Code)
D613-10410

A-1

package Browser_Version_Description is

-- Note: All Browser source code will be located in the directory:
-- geech\$dua0:[incr1.boeing.q12.cdrl410.Browser] on the Boeing Wichita
-- repository after 17-March-1989.

CSCI_Architecture

-- The Ada static architecture of the Browser is depicted in figure 1. There
-- are seven packages and the calling procedure (or driver). Each package
-- encapsulates a group of logically related operations (or subprograms).

-- The compilation order for the units are:

-- STRING_UTILITIES -- Provides operations on strings
-- (in files string_utilities_ada,
-- string_utilities.ada)
--
-- BROWSER_TYPES -- Package containing common type declarations
-- (in files browser_types_ada, Browser_Types.ada)
--
-- MACHINE_OPS -- Package containing machine dependent operations
-- (in files Machine_Ops_ada, Machine_Ops.ada)
--
-- FILE_OPS -- Package containing file i_o operations
-- (in files File_Ops_ada, File_Ops.ada)
--
-- EXTRACT_OPS -- Package containing the operations for the search
-- functions of the Browser (in files Extract_Ops_ada,
-- Extract_Ops.ada)
--
-- MENU_OPS -- Package containing the operations for the menu
-- functions of the Browser (in files Menu_Ops_ada,
-- enu_Ops.ada)
--
-- BROWSER_OPS -- Package containing the executive type operations for
-- the Browser (in files Browser_Ops_ada,
-- Browser_Ops.ada)
--
-- BROWSER_DRIVER -- This is the main calling procedure for the Browser
-- (in file Browser_Driver.ada)

Virtual Interfaces

-- There are no virtual interfaces for this tool.

Tool Interfaces

-- Browser does not interface with any other tool. The Browser interfaces
-- with the user through procedures in TEXT_IO package (ie. get_line,
-- put_line).

Hardware Interfaces

-- The Browser requires an ASCII terminal. This Browser version was developed
-- to operate on a VAX operating system (ver. 4.7). The VAX dependent
-- operations are grouped in the Machine_Ops package and may be easily
-- modified to operate under another operating system.

Functional Control and Data Flow

-- The purpose of the Browser tool is to provide a non-VMS means of viewing
-- VMS directories and VMS files. To navigate through any combination of
-- possible VMS directory structures (directed graphs), the Browser uses the
-- recursion feature of Ada. As the user descends down a directory
-- structure, the Browser will call itself to present the user with a menu
-- formatted display screen of the contents of the current directory. As the
-- user ascends back up the directory structure, the previous directory is
-- redisplayed.

Browser Package Structure

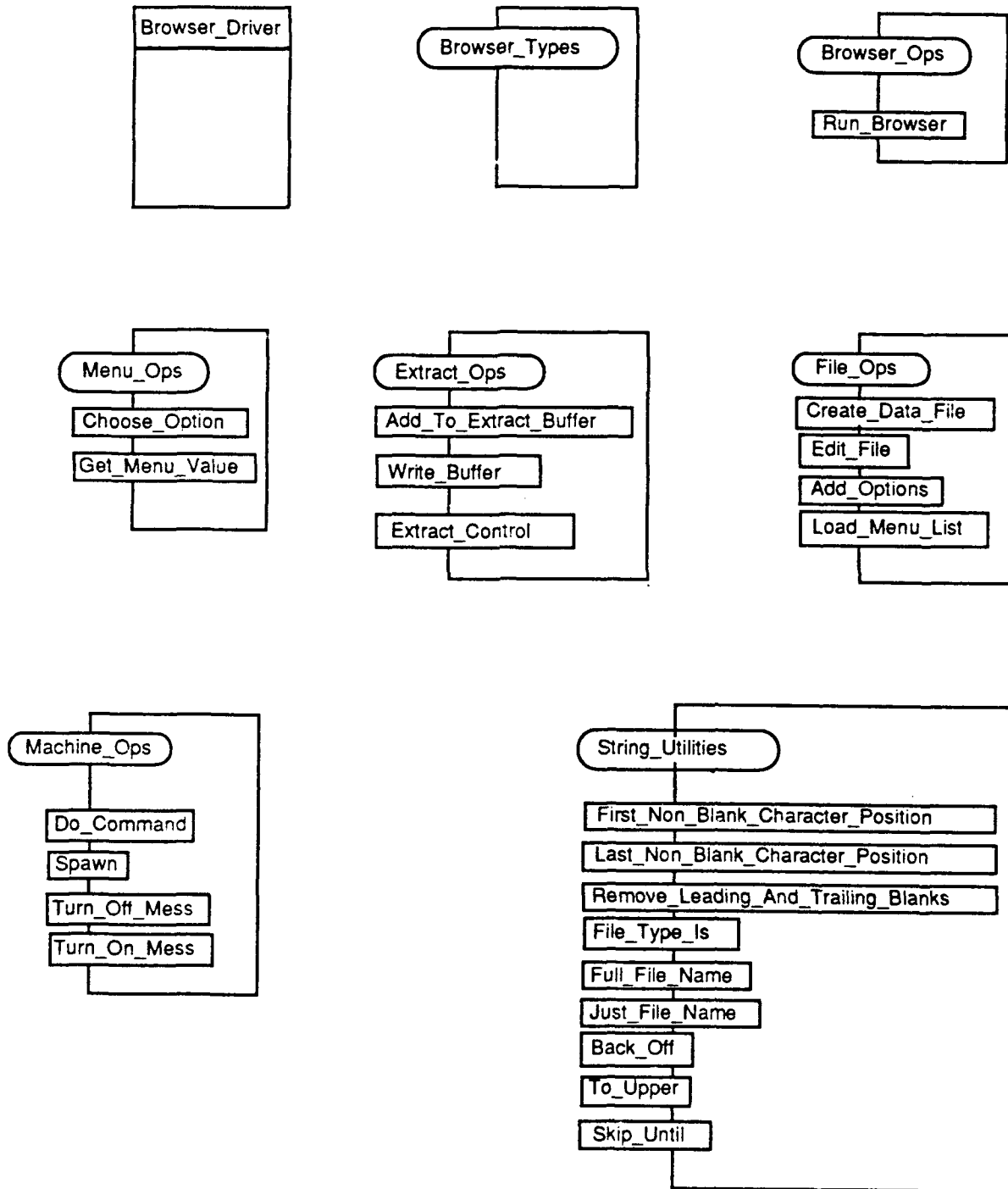


Figure 1

-- The Browser does not move the users VMS directory default position.
-- Instead, the Browser maintains a Virtual Path variable that contains the
-- VMS directory path for the current directory being displayed.

-- The Browser presents the user with a menu formatted screen of a VMS
-- directory by:

- 1) creating a file (in a specified 'utility' directory) that contains a
-- directory listing of the user specified directory
- 2) reading from that file the contents of the directory
- 3) placing each file/subdirectory name into a linked list
- 4) deleting that file in the utility directory

-- The Browser uses the linked list of file/subdirectory names to present
-- these names to the user in a menu format. When the user
-- chooses from the menu a file name, the Browser will allow the user to
-- edit that file. When the user exits the edit session, the same menu is
-- displayed to the user. When the user chooses from the menu a
-- subdirectory, the Browser uses a recursive call to generate a new linked
-- list for that subdirectory. By using the recursion feature in this way,
-- the linked lists of previous directories viewed by the user are not
-- de-referenced, and therefore do not have to be re-generated. This of
-- course is only true for users current directed path, and not for previous
-- directed paths.

-- The Browser does require two files to exist before the Browser will
-- execute. These files are:

-- 1) Menu_1 - This file contains a list of the top level commands
-- that the user may want to perform. An example of this file follows.

-- This is the first menu for the stars foundation
-- Browser tool. ** Make certain that this file remains version 1!
-- (Do not delete these first three lines. Program dependent.)
Review STARS Foundation Projects
Help
or 'X' to Exit (Return to MAIN MENU)
--
-- (Do not delete these last two lines. Program Dependent.)

-- These commands are parsed in the procedure
-- Browser_Ops.Execute_Main_Choice. To change, or add capabilities to this
-- menu, see this procedure.

-- 2) Menu_2 - This file is a listing (dir/col=1) of the top level
-- directory that will be browsed. The purpose of this file is to allow the
-- top level file names, and sub directories to be annotated with a brief
-- explanation of the contents of the file/subdirectory. Here is an example
-- of a Menu_2 file:

--1 This is the second menu for the stars foundation
--2 Browser tool. ** Make certain that this file remains version 1!
--3 (Do not delete these first three lines. Program dependent.)
ADSYSTECH.DIR;1 -Ada Run-Time Support Services (ARTS)
AETECH_VI.DIR;1 -Rapid search and Retrieval (RSR)
BDM.DIR;1 -Secure File Transfer System
CSC.DIR;1 -Transparent_Sequential_IO, & Ada Types Library
--1
--2 (Do not delete these last two lines. Program Dependent.)

-- To create such a file, follow these VMS commands:

-- dir/col=1/output=usr\$dsk:[top_level]menu_2.dat usr\$dsk:[top_level_dir]
-- edit menu_2.dat
-- .. add annotations
-- .. remove any files that you don't want the user to browse at top
-- level

```

--      exit the editor
--      purge usr$dsk:[top_level_dir]menu_2.dat
--      rename usr$dsk:[top_level_dir]menu_2.dat;2 -
--          usr$dsk:[top_level_dir]menu_2.dat;1

```

The procedure that calls the Browser, Browser_Driver.ada, specifies all of the menu names and their locations.
(STLDD 3.4)

System Environment

The Browser was developed in Ada, and targeted to the VAX OS Version 4.7. The Browser does 'with' the following VAX predefined package:

```

LIB  -- Declarations of types, routines and return statuses for the
      General Purpose (LIB) facility of the VAX/VMS Run-Time
      Library.

```

System Parameters

The parameters that the installers/users will supply are found in the procedure Browser_Driver. These parameters are:

```

-----
Top_Level_Dir  -- The top level directory to start browsing.

Menu_1_Title   -- The screen title that you want displayed for menu 1.

Menu_1         -- The complete path name, file name for the first menu to
                be displayed.

Menu_2_Title   -- The screen title that you want displayed for menu 2.

Menu_2         -- The complete path name, file name for the second menu
                to be displayed.

Editor         -- Editor that the user will use.

Utility_Directory -- The directory that will be used for file operations
                    described in the Functional_Control_and_Data_Flow section
                    of this document.
-----

```

Reference CDRL410 for an example of how these parameters are declared and defined.

System Capacities

Browser requires 227 blocks of storage space for the source code. The Browser is not memory intensive.

Installation Instructions

The initial version (1) of the Browser was designed to run on the VAX version 4.7. To install this Browser tool on you system, simply copy the source code to your directory, create an ACS library, and compile and link it. To create the DEC ACS Ada library use a command similar to:

```
$ ACS CREATE LIBRARY USR$DSK:[YOUR_DIR.ADALIB]
```

After setting to this library,

```
$ ACS SET LIBRARY USR$DSK:[YOUR_DIR.ADALIB]
```

users can submit a batch file that has the following commands for compiling the Browser routines.

```

$ ada String_Uutilities_
$ ada String_Uutilities_

```

```

--      $ ada Browser_Types_
--      $ ada Machine_Ops_
--      $ ada Machine_Ops_
--      $ ada File_Ops_
--      $ ada File_Ops_
--      $ ada Extract_Ops_
--      $ ada Extract_Ops_
--      $ ada Menu_Ops_
--      $ ada Menu_Ops_
--      $ ada Browser_Ops_
--      $ ada Browser_Ops_
--      $ ada Browser_Driver
--      $ link Browser_Driver
--      $ exit

```

Before the Browser_Driver can be run, the user must create the two menu files that are required for the Browser to run successfully. See the functional_control_and_data_flow section of this document for instructions on how to create these files.

There is one more file that is used for the 'escape to VMS' function of the Browser. The escape to VMS function allows the user to temporarily visit a VMS session. When the user browses a directory and wants to use a VMS command, they can exit the Browser and be placed into the directory they were viewing. This can be done at any time, and into any directory. Users have available to them all of the valid (non-privileged) VMS commands. They can move freely through the directories available. When the user is ready, they will return to the same Browser screen they saw when they left. To accomplish this operation, the Browser tool calls a VMS command file called VMS.COM. This VMS.COM file will be delivered with the Ada source code in CDRL410. This file may be used as is, or modified to meet additional needs. It does not require any compilation or other preprocessing. It must reside in the 'utilities directory' as defined in the calling argument to the procedure Browser_Driver.

Inventory_of_CSCI_Contents

Unit_Type	Unit_Name	File_Name(s)
procedure	Browser_Driver	Browser_Driver_.ada
package	Browser_Ops	Browser_Ops_.ada, Browser_Ops.ada
package	Browser_Types	Browser_Types_.ada, Browser_Types.ada
package	Extract_Ops	Extract_Ops_.ada, Extract_Ops.ada
package	File_Ops	File_Ops_.ada, File_Ops.ada
package	Machine_Ops	Machine_Ops_.ada, Machine_Ops.ada
package	Menu_Ops	Menu_Ops_.ada, Menu_Ops.ada
package	String_Uutilities	String_Uutilities_.ada, String_Uutilities.ada
VMS command file		VMS.com

Adaptation_Data

Once the two menu files have been created and the Browser_Driver has been compiled and linked, and the VMS.com file is in the 'utility directory', the user need only run the Browser_driver executable.

Lessons_Learned

1) The following routines in the LIB package were very useful for executing a VAX Operating System command from an Ada program:

-- Lib.Spawn - this routine will transfer control to the VAX OS and
-- execute a specified VMS command, then return control back to the Ada
-- program and continue.

-- Lib.Do_Command - this routine will transfer control to the VAX OS and
-- execute a specified VMS command. It will not return to the Ada program.

-- 2) In order to temporarily leave an Ada program, and allow the user to
-- enter a VMS session, the VMS 'inquire' command can be used. The inquire
-- statement can be placed inside a loop to trap users VMS commands. The
-- loop is exited when the users satisfies an 'exit loop conditional'. The
-- following VMS commands are an example of this inquire-in-loop process.
-- (The VMS.COM command file delivered in CDRL410 is based on this example.)

```
-- $ ready:                                !loop id
-- $   inquire/nopunctuation next "Browser $"  !get users VMS command
-- $   if next .eqs. "DONE" then goto isdone   !exit loop when DONE
-- $   if next .eqs. "EXIT" then goto isdone   !exit loop when EXIT
-- $   if next .eqs. "X" then goto isdone      !exit loop when X
-- $!
-- $   define/user_mode/NOLOG sys$input sys$command
-- $   set control=y
-- $   on control_y then continue
-- $   on severe_error then continue
-- $   'next'                                !execute users command
-- $   set nocontrol=y
-- $   goto ready                            !goto ready
-- $ isdone:
-- $   exit                                  !exit this command file
```

-- Revision_History

-- Part Number:
-- Author: Charles Elsner (316) 526-4661
-- Version: 1.0;
-- Change Summary: Initial Release
-- Date: 6-March-89

-- Distribution_and_Copyright

-- This prologue must be included in all copies of this software.
-- This software is released to the Ada community.
-- This software is released to the Public Domain (Note:
-- software released to the Public Domain is not subject
-- to copyright protection).
-- Restrictions on use or distribution: NONE

-- Disclaimer

-- This software and its documentation are provided "AS IS" and
-- without any expressed or implied warranties whatsoever.

-- No warranties as to performance, merchantability, or fitness
-- for a particular purpose exist.

-- Because of the diversity of conditions and hardware under
-- which this software may be used, no warranty of fitness for
-- a particular purpose is offered. The user is advised to test
-- the software thoroughly before relying on it. The user
-- must assume the entire risk and liability of using this
-- software.

-- In no event shall any person or organization or people be
-- held responsible for any direct, indirect, consequential
-- or inconsequential damages or lost profits.

-- end Browser_Version_Description;

--||



package String_Utillities is

```
--| Description
--|   This package contains string operations.
--|
--| Requirements Satisfied
--|   This package does not meet any specific STARS requirements.
--|
--| Exceptions Raised
--|
--|   None.
--|
--| Contact
--|   Charles Elsner
--|   Boeing Military Airplane
--|   3801 S. Oliver (ms K80-13)
--|   Wichita, KS 67277
--|
--| Implementation Dependencies and Assumptions:
--|   Currently, this package uses some of the predefined types from package
--|   Standard.
--|
--| Revision History
--|   Part Number:
--|   Author: Charles Elsner
--|   Version: 1.0;
--|   Change Summary: Initial_Release
--|   Date: 20-FEB-89
--|
--| Distribution and Copyright
--|   This Prologue must be included in all copies of this software.
--|   This software is released to the Ada community.
--|   This software is released to the Public Domain (Note: software released
--|   to the Public Domain is not subject to copyright protection).
--|   Restrictions on use or distribution: NONE
--|
--| Disclaimer
--|   This software and its documentation are provided "AS IS" and
--|   without any expressed or implied warranties whatsoever.
--|
--|   No warranties as to performance, merchantability, or fitness
--|   for a particular purpose exists.
--|
--|   Because of the diversity of conditions and hardware under
--|   which this software may be used, no warranty of fitness for
--|   a particular purpose is offered. The user is advised to test
--|   the software thoroughly before relying on it. The user
--|   must assume the entire risk and liability of using the software.
--|
--|   In no event shall any person or organization of people be
--|   held responsible for any direct, indirect, consequential
--|   or inconsequential damages or lost profits.
```

```
function First_Non_Blank_Character_Position (In_String : string)
return natural;
```

```
--| Description
--|   This function returns the first non blank character position in
--|   a string.
--||
```

```
function Last_Non_Blank_Character_Position (In_String : string)
```

```

                                return natural;
--| Description
--|   This function returns the last non blank character position in
--|   a string.
--||

function Remove_Leading_And_Trailing_Blanks (From_String : string)
                                return string;
--| Description
--|   This function removes leading and trailing blanks from a string.
--||

function File_Type_Is (In_String : string) return string;
--| Description
--|   This function returns the file type of a vms file.
--||

function Full_File_Name (In_String : string) return string;
--| Description
--|   This function returns the complete vms file name less the version.
--||

function Just_File_Name (In_String : string) return string;
--| Description
--|   This file returns the file name minus the type and version.
--||

function Back_Off (In_String : string) return string;
--| Description
--|   This function removes the last directory in a VMS path name.
--|   ex.  New_Path := Back_Off("geech$dua0:[top.next]");
--|         text io.put_line(New_Path);
--|         geech$dua0:[top]
--||

function To_Upper (Input : string) return string;
--| Description
--|   This function returns a string in upper case.

function Skip_Until (Some_String : in string;
                    Until_Char   : in character) return natural;
--| Description
--|   Scan the string until either the end of string is found or the
--|   specified character is found. Return the number of characters
--|   skipped. If the character was not found, return a value of 0.
--||

end String_Uilities;
```

package body String_Uutilities is

--| Description

--| This package contains string operations.

--| Requirements Satisfied

--| This package does not meet any specific STARS requirements.

--| Exceptions Raised

--| None.

--| Contact

--| Charles Elsner

--| Boeing Military Airplane

--| 3801 S. Oliver (ms K80-13)

--| Wichita, KS 67277

--| Implementation Dependencies and Assumptions:

--| Currently, this package uses some of the predefined types from package
--| Standard.

--| Revision History

--| Part Number:

--| Author: Charles Elsner

--| Version: 1.0;

--| Change Summary: Initial_Release

--| Date: 20-FEB-89

--| Distribution and Copyright

--| This Prologue must be included in all copies of this software.

--| This software is released to the Ada community.

--| This software is released to the Public Domain (Note: software released
--| to the Public Domain is not subject to copyright protection).

--| Restrictions on use or distribution: NONE

--| Disclaimer

--| This software and its documentation are provided "AS IS" and
--| without any expressed or implied warranties whatsoever.

--| No warranties as to performance, merchantability, or fitness
--| for a particular purpose exists.

--| Because of the diversity of conditions and hardware under
--| which this software may be used, no warranty of fitness for
--| a particular purpose is offered. The user is advised to test
--| the software thoroughly before relying on it. The user
--| must assume the entire risk and liability of using the software.

--| In no event shall any person or organization of people be
--| held responsible for any direct, indirect, consequential
--| or inconsequential damages or lost profits.

function First_Non_Blank_Character_Position (In_String : string)
return natural is

--| Description

--| This function returns the first non blank character position in
--| a string.

```

    Count : natural;
begin
    for Index in In_String'range loop
        Count := Index;
        exit when In_String (Count) /= ' ';
    end loop;

    return Count;
end First_Non_Blank_Character_Position;

```

```

-----
function Last_Non_Blank_Character_Position (In_String : string)
    return natural is
--| Description
--|   This function returns the last non blank character position in
--|   a string.
--|

```

```

    Count : natural;
begin
    for Index in reverse In_String'range loop
        Count := Index;
        exit when In_String (Count) /= ' ';
    end loop;

    return Count;
end Last_Non_Blank_Character_Position;

```

```

-----
function Remove_Leading_And_Trailing_Blanks (From_String : string)
    return string is
--| Description
--|   This function removes leading and trailing blanks from a string.
--|

```

```

begin
    return From_String
        (First_Non_Blank_Character_Position (From_String) ..
         Last_Non_Blank_Character_Position (From_String));
end Remove_Leading_And_Trailing_Blanks;

```

```

-----
function File_Type_Is (In_String : string) return string is
--| Description
--|   This function returns the file type of a vms file.
--|

```

```

    Count2, Count1 : natural := 0;
begin
    for Index in In_String'range loop
        Count1 := Index;
        exit when In_String(count1) = '.';
    end loop;

    for Index in Count1..In_String'last loop
        Count2 := Index;
        exit when (In_String(Count2) = ';') or (In_String(Count2) = ' ');
    end loop;

    return In_String( (Count1 + 1)..(Count2 - 1));
end File_Type_Is;

```

```

function Full_File_Name (In_String : string) return string is
--| Description
--|   This function returns the complete vms file name less the version.
--||

```

```

    Count1 : natural := 0;
begin
    for Index in In_String'range loop
        Count1 := Index;
        exit when In_String(Count1) = ',';
    end loop;

    return In_String(1..(Count1 - 1));
end Full_File_Name;

```

```

-----
function Just_File_Name (In_String : string) return string is
--| Description
--|   This file returns the file name minus the type and version.
--||

```

```

    Count1 : natural := 0;
begin
    for Index in In_String'range loop
        Count1 := Index;
        exit when In_String(Count1) = '.';
    end loop;

    return In_String(1..(Count1 - 1));
end Just_File_Name;

```

```

-----
function Back_Off ( In_String : string) return string is
--| Description
--|   This function removes the last directory in a VMS path name.
--|   ex.  New_Path := Back_Off("geeche$dua0:[top.next]");
--|       text io.put_line(New_Path);
--|       geeche$dua0:[top]
--||

```

```

    Count1 : natural := 0;

begin
    for Index in reverse In_String'range loop
        Count1 := Index;
        exit when In_String(Count1) = '.';
    end loop;

    return (In_String(In_String'first..(Count1 - 1)) & "]);
end Back_Off;

```

```

-----
function To_Upper (Input : string) return string is
--| Description
--|   This function returns a string in upper case.

```

```

    Temp : string (Input'range) := Input;
    Interval : integer := character'pos('a') - character'pos('A');
begin
    for I in Input'range loop

```

```

        if (Input (i) >= 'a') and (Input(i) <= 'z') then
            Temp(i) := character'val(character'pos(Input(i)) - Interval);
        end if;
    end loop;
    return Temp;
end To_Upper;

```

```

function Skip_Until (Some_String  :  in string;
                    Until_Char   :  in character) return natural is
--| Description
--| Scan the string until either the end of string is found or the
--| specified character is found. Return the number of characters
--| skipped. If the character was not found, return a value of 0.
--||

```

```

    Index : natural := Some_String'first;
    Found : Boolean := false;

```

```

begin
while Index <= Some_String'last loop
    if Some_String(Index) = Until_Char then
        Found := True;
        exit;
    end if;

```

```

    Index := Index + 1;

```

```

end loop;

```

```

if Found then
    return (Index);
else
    return (0);
end if;

```

```

end Skip_Until;

```

```

end String_Uilities;

```



```
with text_io;
with Condition_Handling;
```

```
package Browser_Types is
```

```
-- Description
-- This package contains the type declarations for the Browser operations.
```

```
-- Requirements Satisfied
-- This package does not meet any specific STARS requirements.
```

```
-- Exceptions Raised
```

```
-- None.
```

```
-- Contact
```

```
-- Charles Elsner
-- Boeing Military Airplane
-- 3801 S. Oliver (ms K80-13)
-- Wichita, KS 67277
```

```
-- Implementation Dependencies and Assumptions:
-- Currently, this package uses some of the predefined types from package
-- Standard.
```

```
-- Revision History
```

```
-- Part Number:
-- Author: Charles Elsner
-- Version: 1.0;
-- Change Summary: Initial_Release
-- Date: 20-FEB-89
```

```
-- Distribution and Copyright
```

```
-- This Prologue must be included in all copies of this software.
-- This software is released to the Ada community.
-- This software is released to the Public Domain (Note: software released
-- to the Public Domain is not subject to copyright protection).
-- Restrictions on use or distribution: NONE
```

```
-- Disclaimer
```

```
-- This software and its documentation are provided "AS IS" and
-- without any expressed or implied warranties whatsoever.
```

```
-- No warranties as to performance, merchantability, or fitness
-- for a particular purpose exists.
```

```
-- Because of the diversity of conditions and hardware under
-- which this software may be used, no warranty of fitness for
-- a particular purpose is offered. The user is advised to test
-- the software thoroughly before relying on it. The user
-- must assume the entire risk and liability of using the software.
```

```
-- In no event shall any person or organization of people be
-- held responsible for any direct, indirect, consequential
-- or inconsequential damages or lost profits.
```

```
-- type String_Access_Type is access string;
```

```
type Menu_Line_Type;
```

```
type Menu_List_Type is access Menu_Line_Type;
```

```
type Menu_Line_Type is
record
```

```
Line : String_Access_Type;
Next : Menu_List_Type := null;
end record;

subtype Cond_Value_Type is Condition_Handling.Cond_Value_Type;
Display_Height : constant natural := 24;
Display_Width : constant natural := 80;

--exceptions here

Menu_To_Wide : exception;
File_Cant_Open : exception;

--generics here

package Int_Io is new text_io.integer_io (integer);

end Browser_Types;
```

with Browser_Types;

package Machine_Ops is

--| Description

--| This package contains the visible operations on the object Machine.
--| All of the procedures/functions in this package are machine dependent.
--| They run on the VAX version 4.7.

--| Requirements Satisfied

--| This package does not meet any specific STARS requirements.

--| Exceptions Raised

--| None.

--| Contact

--| Charles Elsner
--| Boeing Military Airplane
--| 3801 S. Oliver (ms K80-13)
--| Wichita, KS 67277

--| Implementation Dependencies and Assumptions:

--| Currently, this package uses some of the predefined types from package
--| Standard.

--| Revision History

--| Part Number:

--| Author: Charles Elsner

--| Version: 1.0;

--| Change Summary: Initial_Release

--| Date: 20-FEB-89

--| Distribution and Copyright

--| This Prologue must be included in all copies of this software.

--| This software is released to the Ada community.

--| This software is released to the Public Domain (Note: software released
--| to the Public Domain is not subject to copyright protection).

--| Restrictions on use or distribution: NONE

--| Disclaimer

--| This software and its documentation are provided "AS IS" and
--| without any expressed or implied warranties whatsoever.

--| No warranties as to performance, merchantability, or fitness
--| for a particular purpose exists.

--| Because of the diversity of conditions and hardware under
--| which this software may be used, no warranty of fitness for
--| a particular purpose is offered. The user is advised to test
--| the software thoroughly before relying on it. The user
--| must assume the entire risk and liability of using the software.

--| In no event shall any person or organization or people be
--| held responsible for any direct, indirect, consequential
--| or inconsequential damages or lost profits.

procedure Do_Command (Status : out Browser_Types.Cond_Value_Type;
In_String : in string);

--| Description

--| This procedure calls the lib.do_command.

--||

```
procedure Spawn (Status      : out Browser_Types.Cond_Value_Type;  
                 In_String  : in string);
```

```
--| Description  
--|   This procedure calls the spawn command.  
--||
```

```
procedure Turn_Off_Mess;
```

```
--| Description  
--|   This command turns off the messages to the screen.  
--||
```

```
procedure Turn_On_Mess;
```

```
--| Description  
--|   This command turns on the messages to the screen.  
--||
```

```
end Machine_Ops;
```

with Lib;

package body Machine_Ops is

-- Description

-- This package contains the hidden operations on the object Machine.
-- All of the procedures/functions in this package are machine dependent.
-- They run on the VAX.

-- Requirements Satisfied

-- This package does not meet any specific STARS requirements.

-- Exceptions Raised

-- None.

-- Contact

-- Charles Elsner
-- Boeing Military Airplane
-- 3801 S. Oliver (ms K80-13)
-- Wichita, KS 67277

-- Implementation Dependencies and Assumptions:

-- Currently, this package uses some of the predefined types from package
-- Standard.

-- Revision History

-- Part Number:
-- Author: Charles Elsner
-- Version: 1.0;
-- Change Summary: Initial_Release
-- Date: 20-FEB-89

-- Distribution and Copyright

-- This Prologue must be included in all copies of this software.
-- This software is released to the Ada community.
-- This software is released to the Public Domain (Note: software released
-- to the Public Domain is not subject to copyright protection).
-- Restrictions on use or distribution: NONE

-- Disclaimer

-- This software and its documentation are provided "AS IS" and
-- without any expressed or implied warranties whatsoever.

-- No warranties as to performance, merchantability, or fitness
-- for a particular purpose exists.

-- Because of the diversity of conditions and hardware under
-- which this software may be used, no warranty of fitness for
-- a particular purpose is offered. The user is advised to test
-- the software thoroughly before relying on it. The user
-- must assume the entire risk and liability of using the software.

-- In no event shall any person or organization or people be
-- held responsible for any direct, indirect, consequential
-- or inconsequential damages or lost profits.

procedure Do_Command (Status : out Browser_Types.Cond_Value_Type;
In_String : in string) is

-- Description

```
--| This procedure calls the lib.do_command.  
--||
```

```
Temp_String : Browser_Types.String_Access_Type := new string'(In_String);  
begin
```

```
    Lib.Do_Command(Status,Temp_String.all);
```

```
end Do_Command;
```

```
-----  
procedure Spawn (Status      : out Browser_Types.Cond_Value_Type;  
                 In_String  : in string) is
```

```
--| Description  
--| This procedure calls the spawn command.  
--||
```

```
Temp_String : browser_Types.String_Access_Type := new string'(In_String);  
begin
```

```
    Lib.Spawn(Status,Temp_String.all);
```

```
end Spawn;
```

```
-----  
procedure Turn_Off_Mess is
```

```
--| Description  
--| This command turns off the messages to the screen.  
--||
```

```
Status : Browser_Types.Cond_Value_Type;
```

```
begin
```

```
    Lib.Spawn(Status,  
              "Set message/noidentification/noseverity/nofacility/notext");
```

```
end;
```

```
-----  
procedure Turn_On_Mess is
```

```
--| Description  
--| This command turns on the messages to the screen.  
--||
```

```
Status : Browser_Types.Cond_Value_Type;
```

```
begin
```

```
    Lib.Spawn(Status,  
              "Set message/identification/severity/facility/text");
```

```
end;
```

```
end Machine_Ops;
```

```
with Text_Io;
with Machine_Ops;
with String_Uutilities;
with Browser_Types;
```

```
package FILE OPS is
```

```
-- Description
```

```
-- This package contains the visible operations for the object File.
```

```
-- Requirements Satisfied
```

```
-- This package does not meet any specific STARS requirements.
```

```
-- Exceptions Raised
```

```
-- None.
```

```
-- Contact
```

```
-- Charles Elsner
-- Boeing Military Airplane
-- 3801 S. Oliver (ms K80-13)
-- Wichita, KS 67277
```

```
-- Implementation Dependencies and Assumptions:
```

```
-- Currently, this package uses some of the predefined types from package
-- Standard.
```

```
-- Revision History
```

```
-- Part Number:
-- Author: Charles Elsner
-- Version: 1.0;
-- Change Summary: Initial_Release
-- Date: 20-FEB-89
```

```
-- Distribution and Copyright
```

```
-- This Prologue must be included in all copies of this software.
-- This software is released to the Ada community.
-- This software is released to the Public Domain (Note: software released
-- to the Public Domain is not subject to copyright protection).
-- Restrictions on use or distribution: NONE
```

```
-- Disclaimer
```

```
-- This software and its documentation are provided "AS IS" and
-- without any expressed or implied warranties whatsoever.
```

```
-- No warranties as to performance, merchantability, or fitness
-- for a particular purpose exists.
```

```
-- Because of the diversity of conditions and hardware under
-- which this software may be used, no warranty of fitness for
-- a particular purpose is offered. The user is advised to test
-- the software thoroughly before relying on it. The user
-- must assume the entire risk and liability of using the software.
```

```
-- In no event shall any person or organization of people be
-- held responsible for any direct, indirect, consequential
-- or inconsequential damages or lost profits.
```

```
procedure Create_Data_File(
```

```
    Utility_Dir      : in Browser_Types.String_Access_Type;
    File_Line        : in Browser_Types.String_Access_Type;
    User              : in Browser_Types.String_Access_Type;
```

```

        Virtual_Path      : in out Browser_Types.String_Access_Type;
        File               : out Browser_Types.String_Access_Type);
--| Description
--| This routine creates the vms file containing the directory listing
--| of vms directory. It also updates the users virtual path (ie. what
--| directory the user is currently viewing.)
--||

procedure Edit_File (
        Virtual_Path      : in Browser_Types.String_Access_Type;
        Editor            : in Browser_Types.String_Access_Type;
        Line              : in Browser_Types.String_Access_Type);
--| Description
--| This routine edits a vms file.
--||

procedure Add_Options (
        File              : in Text_IO.File_Type;
        Ptr               : out Browser_Types.Menu_List_Type;
        Number_Of_Choices : out natural;
        Longest_Menu_Line : out natural);
--| Description
--| This routine reads a vms file and loads them into a linked list.
--||

procedure Load_Menu_List (
        File_Name         : in string;
        Ptr              : out Browser_Types.Menu_List_Type;
        Data_File         : in out Text_IO.File_Type;
        Number_Of_Choices : out natural;
        Longest_Menu_Line : out natural);
--| Description
--| This routine calls the add_options routine after determining the file
--| that contains directory listing.
--||

end File_Ops;

```


package body File_Ops is

-- Description

-- This package contains the internal operations on the object File.

-- Requirements Satisfied

-- This package does not meet any specific STARS requirements.

-- Exceptions Raised

-- None.

-- Contact

-- Charles Elsner

-- Boeing Military Airplane

-- 3801 S. Oliver (ms K80-13)

-- Wichita, KS 67277

-- Implementation Dependencies and Assumptions:

-- Currently, this package uses some of the predefined types from package
-- Standard.

-- Revision History

-- Part Number:

-- Author: Charles Elsner

-- Version: 1.0;

-- Change Summary: Initial_Release

-- Date: 20-FEB-89

-- Distribution and Copyright

-- This Prologue must be included in all copies of this software.

-- This software is released to the Ada community.

-- This software is released to the Public Domain (Note: software released
-- to the Public Domain is not subject to copyright protection).

-- Restrictions on use or distribution: NONE

-- Disclaimer

-- This software and its documentation are provided "AS IS" and
-- without any expressed or implied warranties whatsoever.

-- No warranties as to performance, merchantability, or fitness
-- for a particular purpose exists.

-- Because of the diversity of conditions and hardware under
-- which this software may be used, no warranty of fitness for
-- a particular purpose is offered. The user is advised to test
-- the software thoroughly before relying on it. The user
-- must assume the entire risk and liability of using the software.

-- In no event shall any person or organization or people be
-- held responsible for any direct, indirect, consequential
-- or inconsequential damages or lost profits.

procedure Create_Data_File(
--

Utility_Dir : in Browser_Types.String_Access_Type;

File_Line : in Browser_Types.String_Access_Type;

User : in Browser_Types.String_Access_Type;

Virtual_Path : in out Browser_Types.String_Access_Type;

File : out Browser_Types.String_Access_Type) is

```

--| Description
--|   This routine creates the vms file containing the directory listing
--|   of vms directory.  It also updates the users virtual path (ie. what
--|   directory the user is currently viewing.)
--||

Temp_File : Browser_Types.String_Access_Type;
Status    : Browser_Types.Cond_Value_Type;
File_Name : Browser_Types.String_Access_Type;
Temp_Buff : string(1..80) := (others => ' ');

begin

  -- get file name
  File_Name := new STRING'(String_Uutilities.Just_File_Name(File_Line.all));

  -- prepare to delete the ']'
  Temp_Buff(1..Virtual_Path.all'last) := Virtual_Path.all;

  -- build virtual path
  Virtual_Path := new string'(
    Temp_Buff(1..(Virtual_Path.all'last - 1)) & "." & File_Name.all & "]"");

  -- file will be in the the utility directory
  Temp_File := new string'(Utility_Dir.all & File_Name.all & User.all & ".dat");
  File := Temp_File;

  -- create data file at the utility directory
  Machine_Ops.Spawn(Status,"dir/col=1/output=" &
    Temp_File.all & " " & Virtual_Path.all);

exception
  when others =>
    null;

end Create_Data_File;

-----

procedure Edit_File (
  Virtual_Path : in Browser_Types.String_Access_Type;
  Editor       : in Browser_Types.String_Access_Type;
  Line         : in Browser_Types.String_Access_Type) is

  --| Description
  --|   This routine edits a vms file.
  --||

  Execute_String : Browser_Types.String_Access_Type;
  Status         : Browser_Types.Cond_Value_Type;
  File_Name      : Browser_Types.String_Access_Type;

begin

  -- get file name
  File_Name := new STRING'(String_Uutilities.Full_File_Name(line.all));

  -- build command string
  Execute_String := new string'(Editor.all & "/readonly " & Virtual_Path.all &
    File_Name.all);

  -- edit the file
  Machine_Ops.Spawn(Status,Execute_String.all);

end Edit_File;

```

procedure Add_Options (

File : in Text_Io.File_Type;
Ptr : out Browser_Types.Menu_List_Type;
Number_Of_Choices : out natural;
Longest_Menu_Line : out natural) is

--| Description
--| This routine reads a vms file and loads them into a linked list.
--|

Current : Browser_Types.Menu_List_Type;
Back_Ptr : Browser_Types.Menu_List_Type;
Temp_Longest : natural := 0;
Temp_Choices : natural := 0;
One_Line : Browser_Types.String_Access_Type;
New_String : Browser_Types.String_Access_Type;
Buffer : string(1..Browser_Types.display_width) := (others => ' ');
Test_Buffer : string(1..Browser_Types.display_width) := (others => ' ');
Last_One : natural := 0;
Add_Option_Prob : exception;

begin

Current := new Browser_Types.Menu_Line_Type;
Back_Ptr := Current;
Ptr := Current;

--delete the first three lines
for I in 1..3 loop
if Text_Io.End_Of_File(File) then
Current.Line := new string'("There are no files in this directory.");
Temp_Longest := 37;
Temp_Choices := 1;
else
text_io.skip_line(file => File, spacing => 1);
end if;
end loop;

-- read each line from the file
while not text_io.end_of_file (File) loop

text_io.get_line(file => File,
item => Buffer,
last => Last_One);

-- if this is the second to last line (ie equal to temp_buffer)
if Buffer = Test_Buffer then
Back_Ptr.Next := null;
exit;
end if;

Temp_Choices := Temp_Choices + 1;

One_Line := new string'(
String_Uutilities.Remove_Leading_And_Trailing_Blanks(Buffer));

-- is this the longest one
if Temp_Longest < One_Line.all'length then
if One_Line.all'length > 80 then
Temp_Longest := 80;
else
Temp_Longest := One_Line.all'length;
end if;
end if;

```

Current.Line := new string'(One_Line.all);

if not text_io.end_of_file (File) then
    Current.Next := new Browser_Types.Menu_Line_Type;
    Back_Ptr := Current;
    Current := Current.Next;
    Buffer(1..Buffer'last) := (others => ' ');
end if;

end loop;

Longest_Menu_Line := Temp_Longest;
Number_Of_Choices := Temp_Choices;

exception
    when others =>
        raise Add_Option_Prob;

end Add_Options;

-----

procedure Load_Menu_List (
    File_Name      : in string;
    Ptr            : out Browser_Types.Menu_List_Type;
    Data_File      : in out Text_IO.File_Type;
    Number_Of_Choices : out natural;
    Longest_Menu_Line : out natural) is

    -- Description
    -- This routine calls the add_options routine after determining the file
    -- that contains directory listing.
    --

    Temp_Choices : natural;
    Temp_Longest : natural;
    Temp_Ptr     : Browser_Types.Menu_List_Type;

begin
    -- open the data file
    begin
        text_io.open ( file => Data_File,
                       mode => text_io.in_file,
                       name => File_Name);
    exception
        when others =>
            raise Browser_Types.File_Cant_Open;
    end;

    Add_Options (Data_File, Temp_Ptr, Temp_Choices, Temp_Longest);

    Ptr := Temp_Ptr;
    Number_Of_Choices := Temp_Choices;
    Longest_Menu_Line := Temp_Longest;

end Load_Menu_List;

end File_Ops;

```

with String_Uutilities;
with Browser_Types;

package Extract_Ops is

-- Description

 This package contains the visible operations on the object Extract.

-- Requirements Satisfied

 This package does not meet any specific STARS requirements.

-- Exceptions Raised

 None.

-- Contact

 Charles Elsner
 Boeing Military Airplane
 3801 S. Oliver (ms K80-13)
 Wichita, KS 67277

-- Implementation Dependencies and Assumptions:

 Currently, this package uses some of the predefined types from package
 Standard.

-- Revision History

 Part Number:

 Author: Charles Elsner

 Version: 1.0;

 Change Summary: Initial_Release

 Date: 20-FEB-89

-- Distribution and Copyright

 This Prologue must be included in all copies of this software.

 This software is released to the Ada community.

 This software is released to the Public Domain (Note: software released
 to the Public Domain is not subject to copyright protection).

 Restrictions on use or distribution: NONE

-- Disclaimer

 This software and its documentation are provided "AS IS" and
 without any expressed or implied warranties whatsoever.

 No warranties as to performance, merchantability, or fitness
 for a particular purpose exists.

 Because of the diversity of conditions and hardware under
 which this software may be used, no warranty of fitness for
 a particular purpose is offered. The user is advised to test
 the software thoroughly before relying on it. The user
 must assume the entire risk and liability of using the software.

 In no event shall any person or organization or people be
 held responsible for any direct, indirect, consequential
 or inconsequential damages or lost profits.

-- Procedure Add_To_Extract_Buffer(
 Buffer : in out string;
 Menu_Used : in Browser_Types.Menu_List_Type;
 Displayed : in natural;
 Virtual_Path : in Browser_Types.String_Access_Type);

-- Description

```
--| This procedure will add one to N file names to a buffer (linked list).  
--||
```

```
procedure Write_Buffer;
```

```
--| Description
```

```
--| This procedure writes the buffer to a file in the users directory.  
--||
```

```
procedure Extract_Control;
```

```
--| Description
```

```
--| This procedure is controls the user's interface to the extract  
--| buffer.  
--||
```

```
end Extract_Ops;
```

with Calendar;
with Machine_Ops;
with String_Utilities;
with text_io;
with Browser_Types;
use Browser_Types; -- Using this package to gain visibility to the type
-- declaration operations only;

package body Extract_Ops is

-- Description
-- This package contains the hidden operations on the object Extract.

-- Requirements Satisfied
-- This package does not meet any specific STARS requirements.

-- Exceptions Raised
-- None.

-- Contact
-- Charles Elsner
-- Boeing Military Airplane
-- 3801 S. Oliver (ms K80-13)
-- Wichita, KS 67277

-- Implementation Dependencies and Assumptions:
-- Currently, this package uses some of the predefined types from package
-- Standard.

-- Revision History
-- Part Number:
-- Author: Charles Elsner
-- Version: 1.0;
-- Change Summary: Initial_Release
-- Date: 20-FEB-89

-- Distribution and Copyright
-- This Prologue must be included in all copies of this software.
-- This software is released to the Ada community.
-- This software is released to the Public Domain (Note: software released
-- to the Public Domain is not subject to copyright protection).
-- Restrictions on use or distribution: NONE

-- Disclaimer
-- This software and its documentation are provided "AS IS" and
-- without any expressed or implied warranties whatsoever.
--
-- No warranties as to performance, merchantability, or fitness
-- for a particular purpose exists.
--
-- Because of the diversity of conditions and hardware under
-- which this software may be used, no warranty of fitness for
-- a particular purpose is offered. The user is advised to test
-- the software thoroughly before relying on it. The user
-- must assume the entire risk and liability of using the software.

-- In no event shall any person or organization of people be
-- held responsible for any direct, indirect, consequential
-- or inconsequential damages or lost profits.

```

Head_Ptr      : Browser_Types.Menu_List_Type;
Back_Ptr      : Browser_Types.Menu_List_Type;
Number_in_List : natural;
Longest_Line   : natural;

```

```

use Text_IO;
procedure Help_Extract is
--| Description
--|   This is the help screen for the extract operations.
--||

```

```

    Buffer : string (1..80);
    Length : natural;

```

```

begin

```

```

    -- these io routines are from the Text_IO package

```

```

    new_page;
    put_line("HELP - ");
    new_line;
    put_line("    This is your Selection Buffer.  It contains the ");
    put_line("    files you have choosen from the menus that you wish");
    put_line("    to reuse.  As you exit this Browser Session, your buffer");
    put_line("    will automatically be written to your directory.  The ");
    put_line("    file name will be STARS_BROWSER_current-time.TXT.  You");
    put_line("    should use this file to help you remember what units");
    put_line("    you want to reuse.  If your Buffer is empty, nothing");
    put_line("    will be written to file.  To obtain a copy of your");
    put_line("    selected repository files, the user must use the");
    put_line("    'User Request' option available on the STARS Repository");
    put_line("    menu.");
    new_line;
    put_line("    DE := Delete Entry - If you wish to delete one of the ");
    put_line("    entries from your list, type DE and the number you");
    put_line("    wish to delete.");
    new_line;
    put_line("    AE := Add Entry - To add an entry, type AE, and answer");
    put_line("    the prompt with the file you wish to receive.");
    new_line;
    put("press RETURN for more help");
    get_line (Buffer, Length);
    new_line;
    put_line("    CB := Clear Buffer - To delete all entries in your list");
    put_line("    type CB and return.");
    new_line;
    put_line("    X - This will exit the current menu screen and display the");
    put_line("    previous menu.");
    new_line;
    put_line("    RETURN - by itself will either display the next screen in a");
    put_line("    larger menu, or it will redisplay the same smaller");
    put_line("    menu.  (Larger = more than 18 choices.)");
    new_line;
    put_line("    HELP - This command can be entered any time.  This screen ");
    put_line("    will appear as a result of the help command.");
    new_line;
    new_line;
    put("press RETURN to continue");
    get_line (Buffer, Length);
    new_page;

```

```

end Help_Extract;

```

```

procedure Write_Buffer is

```



```
--| Description
--|   This procedure writes the buffer to a file in the users directory.
--|
```

```
File_Name      : Browser_Types.String_Access_Type;
Current_Ptr    : Browser_Types.Menu_List_Type := Head_Ptr;
Year           : calendar.year_number;
Month          : calendar.month_number;
Day            : calendar.day_number;
Seconds        : calendar.day_duration;
Out_File       : text_io.file_type;
```

```
begin
```

```
-- are there some files in the buffer
if (Number_In_List > 0) and (Head_Ptr /= null) then
```

```
-- what is the time for a unique file name
```

```
calendar.split(
    date      => calendar.clock,
    year      => Year,
    Month     => Month,
    Day       => Day,
    Seconds   => Seconds);
```

```
-- use the seconds to create a unique file name
```

```
File_Name := new string'("SYS$LOGIN:STARS_BROWSER-" &
    integer'image(integer(Seconds)) & ".TXT");
```

```
-- create the file
```

```
text_io.create(
    file => Out_File,
    Mode => text_io.out_file,
    name => File_Name.all);
```

```
-- prolog the file
```

```
text_io.put(Out_File,"This is the list of files/directories that you");
text_io.put_line(Out_File," chose from a ");
text_io.put(Out_File,"STARS_BROWSER session on ");
text_io.put(Out_File,integer'image(Month));
text_io.put(Out_File,"/");
text_io.put(Out_File,integer'image(Day));
text_io.put(Out_File,"/");
text_io.put(Out_File,integer'image(Year));
text_io.put_line(Out_File,".");
text_io.new_line(Out_File,3);
```

```
-- load the file with the buffer
```

```
while Current_Ptr /= null loop
    text_io.put_line(Out_File,Current_Ptr.Line.all);
    text_io.new_line(Out_File);
    Current_Ptr := Current_Ptr.Next;
end loop;
```

```
text_io.close(Out_File);
```

```
text_io.new_line(13);
```

```
text_io.put_line("Your buffer is being written to the file named:");
```

```
text_io.put_line("                " & File_Name.all);
```

```
text_io.new_line(11);
```

```
Delay 3.0;
```

```
end if;
```

```
end Write_Buffer;
```

```

-----
procedure Choose_Option (
    Buffer          : in string;
    Number_Of_Choices : in natural;
    Choice         : out natural;
    Buffer_Status   : out boolean) is
--| Description
--| This routine gives the integer value of the user choice
--||

    Temp_Choice      : integer;
    Length           : natural;
    Integer_Choice    : integer;

begin
    begin
        -- looking for an integer
        Browser_Types.int_io.get (Buffer, Integer_Choice, Length);

    exception
        when others =>
            -- integer within range not entered
            declare
                package Flt_Io is new text_io.float_io (float);
                Float_Choice : float;
            begin
                -- was a float entered?
                flt_io.get (Buffer, Float_Choice, Length);
                -- Convert float to integer
                Integer_Choice := integer (Float_Choice);
            exception
                when others =>
                    -- nothing meaningful entered
                    Integer_Choice := 0;
            end;
        end;

    Temp_Choice := Integer_Choice;

    if not (Temp_Choice <= Number_Of_Choices and Temp_Choice > 0) then
        Buffer_Status := false;
    else
        Buffer_Status := true;
    end if;

    Choice := Temp_Choice;
end Choose_Option;

```

```

-----

Procedure Add (File_Name : in Browser_Types.String_Access_Type) is
--| Description
--| This procedure adds a single filename to linked list.
--||

Current_Ptr : Browser_Types.Menu_List_Type;

begin
    -- when head ptr is null - put the file_name in the first cell
    if Head_Ptr = null then
        Current_Ptr := new Browser_Types.Menu_Line_Type;
        Current_Ptr.Line := new string'(File_Name.all);
        Current_Ptr.Next := null;
    end if;
end Add;

```

```

    Head_Ptr      := Current_Ptr;
    Back_Ptr      := Head_Ptr;

-- else put the file name at the end of the list
else
    Current_Ptr    := new Browser_Types.Menu_Line_Type;
    Current_Ptr.Line := new string'(File_Name.all);
    Current_Ptr.Next := null;
    Back_Ptr.Next   := Current_Ptr;
    Back_Ptr        := Current_Ptr;
end if;

-- Add one to the number in the list
Number_In_List := Number_In_List + 1;

-- is this the longest one
if Longest_line < Current_Ptr.Line.all'length then
    if Current_Ptr.Line.all'length > Browser_Types.Display_Width then
        Longest_Line := Browser_Types.Display_Width;
    else
        Longest_line := Current_Ptr.Line.all'length;
    end if;
end if;

end Add;

-----

procedure Add_To_Extract_Buffer(
    Buffer      : in out string;
    Menu_Used   : in Browser_Types.Menu_List_Type;
    Displayed   : in natural;
    Virtual_Path : in Browser_Types.String_Access_Type) is
--| Description
--| This procedure calls the add routine to add a range of file names to the
--| linked list (ie buffer).
--|
Temp_Choice_1 : natural := 0;
Temp_Choice_2 : natural := 0;
Skip          : natural := 0;
Buffer_Status : boolean := true;
Menu_Ptr      : Browser_Types.Menu_List_Type := Menu_Used;
File_Name     : Browser_Types.String_Access_Type;
Virt_File_Name : Browser_Types.String_Access_Type;
Index         : natural := 1;
Temp_String   : Browser_Types.String_Access_Type;
Vertical_Spacing : integer := 0;

begin

-- remove blanks
Temp_String := new string'(
    String_Uutilities.Remove_Leading_And_Trailing_Blanks(Buffer));

Buffer(1..Temp_String.all'last) := Temp_String.all;

-- find a '--' for a range of text
Skip := String_Uutilities.Skip_Until(Buffer, '--');

-- Range of extractions
if Skip > 0 then
    -- extract choice from buffer
    Choose_Option(
        Buffer      => Buffer(4..(Skip - 1)),
        Number_Of_Choices => Displayed,

```

```

-- verify if buffer choice was in the displayed yet.
if Buffer_Status = true then

    -- get the file from the Menu_List_Type
    for i in 1..(Temp_Choice_1 - 1) loop
        Menu_Ptr := Menu_Ptr.Next;
    end loop;

    -- store the file name
    File_Name :=
        new_string'(String_Uutilities.Full_File_Name(Menu_Ptr.Line.all));
    Virt_File_Name := new_string'(Virtual_Path.all & File_Name.all);

    Add(Virt_File_Name);

    text_io.new_line(12);
    text_io.put_line(File_Name.all & " added to buffer.");
    text_io.new_line(12);
    delay 2.0;
    text_io.new_line(12);

end if;
end if;

-- buffer choice was not valid
if Buffer_Status = false then
    Temp_String := new
        string'(String_Uutilities.Remove_Leading_And_Trailing_Blanks(Buffer));

    text_io.new_line(12);
    text_io.put("Sorry... " & Temp_String.all);
    text_io.put_line(" is not a valid command.");
    text_io.put_line("Type HELP for an explanation of the valid commands.");
    text_io.new_line(12);
    delay 2.0;
    text_io.new_line(12);
end if;

end Add_To_extract_Buffer;

```

```

-----
Procedure Delete (Entry_Num      : in Natural;
                  Last_Current   : in out Browser_Types.Menu_List_Type ) is
--| Description
--|   This procedure will delete a link from the buffer.
--|

```

```

Temp_Back      : Browser_Types.Menu_List_Type := Head_Ptr;
Current_Ptr     : Browser_Types.Menu_List_Type := Head_Ptr;

```

```

begin

```

```

    for i in 1..(Entry_Num - 1) loop
        Temp_Back := Current_Ptr;
        Current_Ptr := Current_Ptr.Next;
    end loop;

```

```

    if Current_Ptr = Last_Current then
        if Number_In_List = 1 then
            Head_Ptr := Null;
            Back_Ptr := Head_Ptr;
            Last_Current := Head_Ptr;
        else
            if Current_Ptr = Head_Ptr then

```

```

        Choice          => Temp_Choice_1,
        Buffer_Status    => Buffer_Status);

if Buffer_Status = true then
    Choose_Option(
        Buffer           => Buffer((Skip + 1)..Buffer'last),
        Number_Of_Choices => Displayed,
        Choice          => Temp_Choice_2,
        Buffer_Status    => Buffer_Status);
end if;

if Buffer_Status = true then
    if Temp_Choice_2 >= Temp_Choice_1 then

        if (Temp_Choice_2 - Temp_Choice_1) > 18 then
            Vertical_Spacing := 1;
        else
            Vertical_Spacing := (Browser.Types.Display_Height -
                (Temp_Choice_2 - Temp_Choice_1) - 4) / 2;
        end if;

        for i in 1..(Temp_Choice_1 - 1) loop
            Menu_Ptr := Menu_Ptr.Next;
        end loop;

        -- store the file name
        File_Name :=
            new_string'(String.Utilities.Full_File_Name(Menu_Ptr.Line.all));
        Virt_File_Name := new_string'(Virtual_Path.all & File_Name.all);

        text_io.new_line(text_io.count (Vertical_Spacing + 8));
        Add(Virt_File_Name);

        text_io.put_line(File_Name.all & " added to buffer.");

        for i in 1..(Temp_Choice_2 - Temp_Choice_1) loop

            Menu_Ptr := Menu_Ptr.Next;

            -- store the file name
            File_Name :=
                new_string'(String.Utilities.Full_File_Name(Menu_Ptr.Line.all));
            Virt_File_Name := new_string'(Virtual_Path.all & File_Name.all);

            Add(Virt_File_Name);

            text_io.put_line(File_Name.all & " added to buffer.");

        end loop;

        text_io.new_line (text_io.count (Vertical_Spacing));
        delay 2.0;
    else
        Buffer_Status := false;
    end if;

end if;

else
    -- extract choice from buffer
    Choose_Option(
        Buffer           => Buffer(4..Buffer'last),
        Number_Of_Choices => Displayed,
        Choice          => Temp_Choice_1,
        Buffer_Status    => Buffer_Status);

```

```

        Head_Ptr := Head_Ptr.Next;
        Last_Current := Head_Ptr;
    elsif Current_Ptr = Back_Ptr then
        Back_Ptr := Temp_Back;
        Last_Current := Back_Ptr;
    else
        Temp_Back.next := Current_Ptr.next;
    end if;
end if;
else
    if Number_In_List = 1 then
        Head_Ptr := Null;
        Back_Ptr := Head_Ptr;
    else
        if Current_Ptr = Head_Ptr then
            Head_Ptr := Head_Ptr.Next;
        elsif Current_Ptr = Back_Ptr then
            Back_Ptr := Temp_Back;
            Back_Ptr.Next := null;
        else
            Temp_Back.next := Current_Ptr.next;
        end if;
    end if;
end if;

```

```

-- Subtract one to the number in the list
Number_In_List := Number_In_List - 1;

-- If buffer is empty, zero the longest line
if Number_In_List = 0 then
    Longest_Line := 0;
end if;

```

end Delete;

```

-----
procedure Delete_From_Extract_Buffer(
    Buffer          : in string;
    Last_Current   : in out Browser_Types.Menu_List_Type;
    Displayed      : in natural) is
--| Description
--| This procedure calls the delete routine to delete a link from the list.
--|

```

```

Temp_Choice      : natural := 0;
Buffer_Status    : boolean := true;
Temp_String      : Browser_Types.String_Access_Type;

```

begin

```

-- extract choice from buffer
Choose_Option(
    Buffer          => Buffer(4..Buffer'last),
    Number_Of_Choices => Displayed,
    Choice         => Temp_Choice,
    Buffer_Status   => Buffer_Status);

```

```

-- verify if buffer choice was in the displayed yet.
if Buffer_Status = true then

```

```

    Delete(
        Entry_Num      => Temp_Choice,
        Last_Current   => Last_Current);

```

```

-- buffer choice was not valid

```

```

else
    Temp_String := new
        string'(String_Uutilities.Remove_Leading_And_Trailing_Blanks(Buffer));

    text_io.new_line(12);
    text_io.put("Sorry... " & Temp_String.all);
    text_io.put_line(" is not a valid command.");
    text_io.put_line("Type HELP for an explanation of the valid commands.");
    text_io.new_line(12);
    delay 2.0;
    text_io.new_line(12);
end if;
end Delete_From_Extract_Buffer;

```

```

-----

procedure Add_Entry_To_Extract_Buffer is
--| Description
--|   This procedure adds a user entry to the list from the key board
--|   and not from the list of files.
--||

```

```

    Buffer          : String (1..256) := (others => ' ');
    Length          : natural;
    Temp_Buffer     : Browser_Types.String_Access_Type;
    Buffer_Problem  : Boolean := False;

```

```

begin

```

```

    text_io.new_line;
    text_io.put("New Entry => ");

```

```

    begin
        text_io.get_line (Buffer, Length);
        exception
            when others =>
                Buffer_Problem := true;
    end;

```

```

    If Buffer_Problem then
        text_io.new_line(12);
        text_io.put_line(" Sorry ... Only 256 Characters Please.");
        text_io.new_line(12);
        delay 2.0;
        text_io.new_line(12);
    else

```

```

        Temp_Buffer := new
            string'(String_Uutilities.Remove_Leading_And_Trailing_Blanks(Buffer));

        Add(Temp_Buffer);
    end if;

```

```

end Add_Entry_To_Extract_Buffer;

```

```

-----

procedure Clear_Buffer( Last_Current : in out Browser_Types.Menu_List_Type) is
--| Description
--|   This procedure clears the users buffer.
--||

```

```

    Buffer : string (1..80) := (others => ' ');
    Length : natural;

```

```

begin

```

```

-- exit the menu
elsif Buffer(1) = 'X' then
    Repeat_Screen := False;
    More := false;
    text_io.new_line(1);

-- delete entry
elsif Buffer(1..2) = "DE" then
    Delete_From_Extract_Buffer(
        Buffer => Buffer,
        Last_Current => Last_Current,
        Displayed => Displayed);

    if Number_In_List = 0 then
        More := false;
    else
        Repeat_Screen := true;
        More := true;
    end if;
    text_io.new_line(1);

-- add entry
elsif Buffer(1..2) = "AE" then
    Add_Entry_To_Extract_Buffer;
    Repeat_Screen := true;
    More := true;
    text_io.new_line(1);

-- clear buffer
elsif Buffer(1..2) = "CB" then
    Clear_Buffer(Last_Current => Last_Current);
    if Number_In_List = 0 then
        More := false;
    else
        More := True;
        Repeat_Screen := True;
    end if;
    text_io.new_line(1);

-- print help screen
elsif Buffer(1..4) = "HELP" then
    Help_Extract;
    Repeat_Screen := true;
    More := true;

-- exit the browser
elsif Buffer(1..4) = "EXIT" then
    Write_Buffer;
    Machine_Ops.Do_Command(Status, "$");

else
    -- get integer value
    Choose_Option(Buffer,
        Displayed,
        Choice,
        Buffer_Status);

    -- user entered invalid command
    if Buffer_Status = false then

        -- user entered invalid command
        text_io.new_line(12);
        text_io.put_line(Buffer(1..length) & " is not valid.");
        text_io.put_line("Type HELP for a list of valid commands.");
        text_io.new_line(12);
    end if;
end if;

```



```

text_io.new_line;
text_io.put("Are You Sure You Want To Clear This Buffer (y) :");
text_io.get_line (Buffer, Length);

if (Buffer(1) = ' ') or (Buffer(1) = 'y') or (Buffer(1) = 'Y') then
  Head_Ptr := Null;
  Back_Ptr := Head_Ptr;
  Last_Current := Head_Ptr;
  Number_In_List := 0;
  Longest_Line := 0;
end if;

end Clear_Buffer;

```

```

-----
procedure Get_Menu_Value ( Choice : out natural) is

```

```

-- Description
-- This procedure will display a menu with the given title,
-- all of which is centered on the screen. A prompt will be
-- given to the user to enter a choice. If an illegal choice
-- is entered, the menu will be redrawn and the prompt will
-- be repeated. Control will return only when a legal value
-- is entered.
--|

Vertical_Spacing      : integer := 0;
Lines                 : boolean := true;
More                  : boolean := true;
Current               : Browser_Types.Menu_List_Type := Head_Ptr;
Last_Current          : Browser_Types.Menu_List_Type := Head_Ptr;
Repeat_Screen         : boolean := false;
Index                 : natural := 0;
Displayed              : natural := 0;
Last_Displayed        : natural := 0;
Buffer                : string(1..Browser_Types.display_width) :=
                      (others => ' ');
Number_In_List_Title  : Browser_Types.String_Access_Type;
Title_Spacing         : integer;
Choice_Title_Spacing  : integer;
Buffer_Status         : boolean;
Temp_Number_Of_Choices : natural := Number_In_List;
Length                : natural;
Title                 : Browser_Types.String_Access_Type :=
                      new string("Select Buffer");
Buffer_Empty          : String(1..16) := "Buffer is Empty.";
Status                : Browser_Types.Cond_Value_Type;

```

```

begin --get_menu_value

```

```

  -- preset Choice to zero
  Choice := 0;

```

```

  -- display the menu
  while More loop

```

```

    if Current = null then
      Last_Current := Head_Ptr;
      Current := Head_Ptr;
      Displayed := 0;
      Last_Displayed := 0;
    end if;

```

```

  -- reset the last page after a help screen

```

```

if Repeat_Screen then

    Current := Last_Current;
    Displayed := Last_Displayed;
    Repeat_Screen := false;

    -- determine vertical spacing between title and menu and then menu
    -- and prompt
    if Number_In_List > 18 then
        Vertical_Spacing := 1;
    else
        Vertical_Spacing := (Browser_Types.Display_Height -
                               Number_In_List - 4) / 2;
    end if;

    text_io.new_line(12);
    -- if there will be more than one screen, tell user how many
    -- entries
    if (Displayed <= 1) then

        -- put the title out
        Title_Spacing := (Browser_Types.Display_Width -
                           Title.all'length) / 2;
        text_io.set_col (text_io.positive_count (Title_Spacing));
        text_io.put_line (Title.all);

        if Number_In_List > 18 then
            Number_In_List_Title := new string("(" &
                Integer'image(Number_In_List) & " Items.");
            Choice_Title_Spacing := (Browser_Types.Display_Width -
                                     Number_In_List_Title.all'length) / 2;
            text_io.set_col (text_io.positive_count
                               (Choice_Title_Spacing));
            text_io.put_line (Number_In_List_Title.all);
        end if;

    end if;

    text_io.new_line (text_io.count (Vertical_Spacing));

else
    Last_Current := Current;
    Last_Displayed := Displayed;

    text_io.new_line(12);

    -- put the title out
    Title_Spacing := (Browser_Types.Display_Width -
                       Title.all'length) / 2;
    text_io.set_col (text_io.positive_count (Title_Spacing));
    text_io.put_line (Title.all);

    -- determine vertical spacing between title and menu and then menu
    -- and prompt
    if Number_In_List > 18 then
        Vertical_Spacing := 1;
    else
        Vertical_Spacing := (Browser_Types.Display_Height -
                               Number_In_List - 4) / 2;
    end if;

    -- if there will be more than one screen, tell user how many
    -- entries
    if Number_In_List > 18 then
        Number_In_List_Title := new string("(" &
            integer'image(Number_In_List) & " Items.");
    end if;

```

```

        Choice_Title_Spacing := (Browser_Types.Display_Width -
        Number_In_List_Title.all'length) / 2;
        text_io.set_col (text_io.positive_count (Choice_Title_Spacing));
        text_io.put_line (Number_In_List_Title.all);
    end if;

    text_io.new_line (text_io.count (Vertical_Spacing));

end if;

if Current = null then
    text_io.set_col(text_io.positive_count(
        ((Browser_Types.Display_Width + 2) - Buffer_Empty'length)/2));
    text_io.put_line(Buffer_Empty);
else
    Index := 0;
    Lines := True;

    while Lines and (Index <= (Browser_Types.Display_Height - 7)) Loop
        Index := Index + 1;

    begin
        -- set column for longest menu line
        text_io.set_col(text_io.positive_count(
            ((Browser_Types.Display_Width + 2) - Longest_Line) / 2));

        -- line up the menu choices better
        if (Displayed + Index) in 1..9 then
            text_io.put(' ' & natural'image(Displayed + Index));
        else
            text_io.put(natural'image(Displayed + Index));
        end if;

        text_io.put(" ");
        text_io.put_line (Current.Line.all);
        Current := Current.Next;
        if Current = null then
            Lines := false;
        end if;

    exception
        when others =>
            Lines := false;

    end;
end Loop; -- lines

    Displayed := Displayed + Index;
end if;

text_io.new_line (text_io.count (Vertical_Spacing));

text_io.put(" [ DE # (Delete Entry), AE (Add Entry),");
text_io.put_line(" CB (Clear_Buffer), X (Prev_Menu), HELP ]");
text_io.put("=> ");

text_io.get_line (Buffer, Length);

-- convert to upper case
Buffer := String_Uutilities.To_Upper(Buffer);

-- does the user want to continue the menu
if Buffer(1) = ' ' then
    Repeat_Screen := False;
    More := true;
    text_io.new_line(1);

```

```

        delay(2.0);

        -- recursive call to get_menu_value
        Get_Menu_Value (Choice);
    end if;
    More := false;

end if;

Buffer(1..Buffer'last) := (others => ' ');

end loop; -- more loop

exception
    when others =>
        null;

end Get_Menu_Value;

-----

procedure Extract_Control is
--| Description
--|   This is the procedure that presents a user interface for extract buffer
--|   operations (other than the add_to_extract_buffer which is called from the
--|   browser menu.
--||

Title   : Browser_Types.String_Access_Type := new string'("Select Buffer");
Choice  : natural := 0;

begin

    extract : loop

        --Display and get menu value
        text_io.new_line(1);      -- to make a clean screen
        Get_Menu_Value( Choice => Choice);

    exit Extract when Choice = 0;
    end loop Extract;

    text_io.new_line(24);

end Extract_Control;

end Extract_Ops;

```

with Browser_Types;

package Menu_Ops is

--| Description

--| This package contains the visible operations on the object Menu.

--| Requirements Satisfied

--| This package does not meet any specific STARS requirements.

--| Exceptions Raised

--| None.

--| Contact

--| Charles Elsner
--| Boeing Military Airplane
--| 3801 S. Oliver (ms K80-13)
--| Wichita, KS 67277

--| Implementation Dependencies and Assumptions:

--| Currently, this package uses some of the predefined types from package
--| Standard.

--| Revision History

--| Part Number:
--| Author: Charles Elsner
--| Version: 1.0;
--| Change Summary: Initial_Release
--| Date: 20-FEB-89

--| Distribution and Copyright

--| This Prologue must be included in all copies of this software.
--| This software is released to the Ada community.
--| This software is released to the Public Domain (Note: software released
--| to the Public Domain is not subject to copyright protection).
--| Restrictions on use or distribution: NONE

--| Disclaimer

--| This software and its documentation are provided "AS IS" and
--| without any expressed or implied warranties whatsoever.

--| No warranties as to performance, merchantability, or fitness
--| for a particular purpose exists.

--| Because of the diversity of conditions and hardware under
--| which this software may be used, no warranty of fitness for
--| a particular purpose is offered. The user is advised to test
--| the software thoroughly before relying on it. The user
--| must assume the entire risk and liability of using the software.

--| In no event shall any person or organization or people be
--| held responsible for any direct, indirect, consequential
--| or inconsequential damages or lost profits.

procedure Help_Screen;

--| Description

--| Display the help screen.

```

procedure Choose_Option (
    Buffer          : in string;
    Number_Of_Choices : natural;
    Choice         : out integer;
    Buffer_Status   : out boolean);
--| Description
--| This routine will determine the integer value choice from the string
--| buffer.
--|

```

```

procedure Get_Menu_Value (
    Menu_Used       : in Browser_Types.Menu_List_Type;
    Number_Of_Choices : in natural;
    Longest_Menu_Line : in natural;
    Title           : in Browser_Types.String_Access_Type;
    Virtual_Path     : in Browser_Types.String_Access_Type;
    Choice          : out integer);
--| Description
--| This routine will display a menu to the user, which is a linked list
--| of all the files in a directory, gets a response fro the user and
--| passes the reponse to the calling unit.
--|

```

```

end Menu_Ops;

```

```
with Browser_Types;
use Browser_Types; -- Using this package to gain visibility to the type
                    -- declaration operations only;

with Machine_Ops;
with Extract_Ops;
with String_Uutilities;
with Text_IO;
```

```
package body Menu_Ops is
```

```
-- Description
--   This package contains the hidden operations on the object Menu.
```

```
-- Requirements Satisfied
--   This package does not meet any specific STARS requirements.
```

```
-- Exceptions Raised
--   None.
```

```
-- Contact
--   Charles Elsner
--   Boeing Military Airplane
--   3801 S. Oliver (ms K80-13)
--   Wichita, KS 67277
```

```
-- Implementation Dependencies and Assumptions:
--   Currently, this package uses some of the predefined types from package
--   Standard.
```

```
-- Revision History
--   Part Number:
--   Author: Charles Elsner
--   Version: 1.0;
--   Change Summary: Initial_Release
--   Date: 20-FEB-89
```

```
-- Distribution and Copyright
--   This Prologue must be included in all copies of this software.
--   This software is released to the Ada community.
--   This software is released to the Public Domain (Note: software released
--   to the Public Domain is not subject to copyright protection).
--   Restrictions on use or distribution: NONE
```

```
-- Disclaimer
--   This software and its documentation are provided "AS IS" and
--   without any expressed or implied warranties whatsoever.
```

```
--   No warranties as to performance, merchantability, or fitness
--   for a particular purpose exists.
```

```
--   Because of the diversity of conditions and hardware under
--   which this software may be used, no warranty of fitness for
--   a particular purpose is offered. The user is advised to test
--   the software thoroughly before relying on it. The user
--   must assume the entire risk and liability of using the software.
```

```
--   In no event shall any person or organization of people be
--   held responsible for any direct, indirect, consequential
--   or inconsequential damages or lost profits.
```

```
-----
```

```

use text_io;
procedure Help_Screen is
--| Description
--|   Display the help screen.
--|

```

```

    Buffer : string (1..80);
    Length : natural;

```

```

begin

```

```

    -- all of these io routines are from the Text_Io package

```

```

    new_page;
    put_line("HELP - ");
    put_line("  This Directory Browser allows users to examine directories");
    put_line("  and files available on the STARS Repository.  Note: These");
    put_line("  files have not passed the acceptance criteria.");
    put_line("  The commands available are:");
    new_line;
    put_line("  # - The user chooses a number displayed on the current");
    put_line("  menu.  If the number corresponds to a Directory (ex.");
    put_line("  Math Lib.dir), a new menu will be entered which");
    put_line("  displays the files in that directory.  The Directory");
    put_line("  structure can be likened to a directed graph.  The");
    put_line("  user descends by typing the menu number of a directory,");
    put_line("  and ascends by typing 'x'.  If the user chooses");
    put_line("  a number which corresponds to a file (ex. SIN.ADA,");
    put_line("  control will be moved to an editing session for");
    put_line("  that file.  The current editor is the TPU editor");
    put_line("  in READ ONLY mode.  To get help in this editor, just");
    put_line("  type help at the command line, (the command line for");
    put_line("  VAX Terminals is the DO key.)  All floating point");
    put_line("  numbers entered will be truncated.");
    new_line;
    put("press RETURN for more help");
    get_line (Buffer, Length);
    new_line;
    new_line;
    new_line;
    new_line;
    put_line("  SE # - This is the SELECT command.  You can write the");
    put_line("  file name you choose to a buffer. (ex. SE 1, will");
    put_line("  write the file name at the menu option 1 to a");
    put_line("  buffer.)  You may specify a range of files by placing");
    put_line("  a '-' between your range. (ex. SE 1-5, will write the");
    put_line("  file names from menu options 1 through 5 to a buffer.");
    put_line("  SELECT records the file name ONLY.  To review your");
    put_line("  buffer, use the R(EVIEW) command.  To obtain a copy");
    put_line("  of the selected repository files, the user must use the");
    put_line("  'User Request' option available on the STARS Repository");
    put_line("  menu.");
    new_line;
    put_line("  REVIEW - This will review your SELECTION buffer, (the");
    put_line("  files you have selected from the menus that you wish");
    put_line("  to reuse).  As you exit this Browser Session, your buffer");
    put_line("  will automatically be written to your directory.  The");
    put_line("  file name will be STARS_BROWSER_current-time.TXT.  You");
    put_line("  should use this file to help you remember what units");
    put_line("  you want to reuse.  If your Buffer is empty, nothing");
    put_line("  will be written to file.");
    new_line;
    put("press RETURN to continue");
    get_line (Buffer, Length);
    new_line;
    new_line;
    new_line;

```



```

new_line;
put_line(" X - This will exit the current menu screen and display the");
put_line(" previous menu.");
new_line;
put_line(" RETURN - This will either display the next screen in a");
put_line(" large menu, or it will redisplay the current menu.");
new_line;
put_line(" HELP - This command can be entered any time. This screen ");
put_line(" will appear as a result of the help command.");
new_line;
put_line(" EXIT or ctrl-z - To exit the Browser Session, type EXIT");
put_line(" or ctrl-z at the prompt. Exit will write your Select");
put_line(" Buffer to file, ctrl-z will not.");
new_line;
put_line(" MENU - This command will go to first Browser menu.");
new_line;
put_line(" VMS - This will exit the Browser and place you into the VMS");
put_line(" directory you were browsing. Once in the VMS directory,");
put_line(" you can do any normal VMS operations. Type 'X' to return");
put_line(" to the browser.");
new_line;
put("press RETURN to continue");
get_line(Buffer, Length);
new_page;

```

end Help_Screen;

```

-----
procedure Choose_Option (
    Buffer          : in string;
    Number_Of_Choices : natural;
    Choice         : out integer;
    Buffer_Status   : out boolean) is
--| Description
--| This routine will determine the integer value choice from the string
--| buffer.
--|

```

```

    Temp_Choice      : integer;
    Length           : natural;
    Integer_Choice    : integer;

```

```

begin
    begin
        -- looking for an integer
        Browser_Types.int_io.get (Buffer, Integer_Choice, Length);

    exception
        when others =>
            -- integer within range not entered
            declare
                package Flt_Io is new text_io.float_io (float);
                Float_Choice : float;
            begin
                -- was a float entered?
                flt_io.get (Buffer, Float_Choice, Length);
                -- Convert float to integer
                Integer_Choice := integer (Float_Choice);
            exception
                when others =>
                    -- nothing meaningful entered
                    Integer_Choice := 0;
            end;
        end;
    end;
end;

```

```

Temp_Choice := Integer_Choice;
-- did you find a valid integer.
if not (Temp_Choice <= Number_Of_Choices and Temp_Choice > 0) then
    Buffer_Status := false;
else
    Buffer_Status := true;
end if;

Choice := Temp_Choice;
end Choose_Option;

```

```

-----
procedure Get_Menu_Value (
    Menu_Used       : in Browser_Types.Menu_List_Type;
    Number_Of_Choices : in natural;
    Longest_Menu_Line : in natural;
    Title           : in Browser_Types.String_Access_Type;
    Virtual_Path    : in Browser_Types.String_Access_Type;
    Choice          : out integer) is

```

```

-- Description
-- This procedure will display a menu with the given title
-- (all of which is centered on the screen). A prompt will be
-- given to the user to enter a choice. If an illegal choice
-- is entered, the menu will be redrawn and the prompt will
-- be repeated. Control will return only when a legal value
-- is entered.
--

```

```

Vertical_Spacing      : integer := 0;
Lines                 : boolean := true;
More                  : boolean := true;
Current               : Browser_Types.Menu_List_Type := Menu_Used;
Last_Current          : Browser_Types.Menu_List_Type := Menu_Used;
Repeat_Screen         : boolean := false;
Index                 : natural := 0;
Displayed             : natural := 0;
Last_Displayed        : natural := 0;
Buffer                : string(1..Browser_Types.display_width) :=
                        (others => ' ');
Number_Of_Choices_Title : Browser_Types.String_Access_Type;
Title_Spacing         : integer;
Choice_Title_Spacing   : integer;
Buffer_Status         : boolean;
Temp_Number_Of_Choices : natural := Number_of_Choices;
Length                : natural;
Status                : Browser_Types.Cond_Value_Type;

```

```

begin --get_menu_value

```

```

-- prepare title for number of choices the user will have
Number_Of_Choices_Title := new string'("(" &
    integer'image(Number_Of_Choices) & " Choices.)");

```

```

if Longest_Menu_Line > Browser_Types.Display_Width then
    raise Browser_Types.Menu_To_Wide;
end if;

```

```

-- determine vertical spacing between title and menu and then menu
-- and prompt
if Number_Of_Choices > 18 then
    Vertical_Spacing := 1;
else

```

```

    Vertical_Spacing := (Browser_Types.Display_Height -
                          Number_Of_Choices - 4) / 2;
end if;

text_io.new_line(24);

-- put the title out
Title_Spacing := (Browser_Types.Display_Width - title.all'length) / 2;
Choice_Title_Spacing := (Browser_Types.Display_Width -
                          Number_Of_Choices_Title.all'length) / 2;
text_io.set_col (text_io.positive_count (Title_Spacing));
text_io.put_line (Title.all);

-- if there will be more than one screen of choices, tell user how many
if Number_Of_Choices > 18 then
    text_io.set_col (text_io.positive_count (Choice_Title_Spacing));
    text_io.put_line (Number_Of_Choices_Title.all);
end if;

text_io.new_line (text_io.count (Vertical_Spacing));

--preset choice to zero
Choice := 0;

-- display the menu
while More loop

    -- reset the last page after a help screen
    if Repeat_Screen then
        Current := Last_Current;
        Displayed := Last_Displayed;
        Repeat_Screen := false;
        Lines := true;
        text_io.set_col (text_io.positive_count (Title_Spacing));
        text_io.put_line (Title.all);
        if (Displayed <= 1) and (Number_Of_Choices > 18) then
            text_io.set_col (text_io.positive_count (Choice_Title_Spacing));
            text_io.put_line (Number_Of_Choices_Title.all);
        end if;
        text_io.new_line (text_io.count (Vertical_Spacing));
    else
        Last_Current := Current;
        Last_Displayed := Displayed;
    end if;

    Index := 0;
    Lines := True;

    while Lines and (Index <= (Browser_Types.Display_Height - 7)) Loop
        Index := Index + 1;

        begin
            -- set column for longest menu line
            text_io.set_col(text_io.positive_count(
                ((Browser_Types.Display_Width + 2) - Longest_Menu_Line)/2));

            -- line up the menu choices better
            if (Displayed + Index) in 1..9 then
                text_io.put(' ' & natural'image(Displayed + Index));
            else
                text_io.put(natural'image(Displayed + Index));
            end if;

            text_io.put(" ");
            text_io.put_line (Current.Line.all);
            Current := Current.Next;
        end
    end while
end while

```

```

        if Current = null then
            Lines := false;
        end if;

    exception
        when others =>
            Lines := false;
    end;
end Loop;

Displayed := Displayed + Index;

-- output prompt
begin
    text_io.new_line (text_io.count (Vertical_Spacing));
exception
    when constraint_error =>
        null;
end;

text_io.put( "[ X (Previous Menu), # (Menu Select), ";
text_io.put_line("SE # (Select No.), R(EVIEW Buffer), HELP ]");
text_io.put( "[ hit RETURN (continue), VMS (exit to VMS), ";
text_io.put("MENU (1st menu)] Command => ");

text_io.get_line (Buffer, Length);

-- convert to upper case
Buffer := String_Uutilities.To_Upper(Buffer);

-- does the user want to continue the menu
if (Buffer(1) = ' ') and (Displayed < Number_Of_Choices) then
    More := true;
    Repeat_Screen := false;

-- exit the menu
elsif Buffer(1) = 'X' then
    More := false;

-- exit the browser
elsif Buffer(1..4) = "EXIT" then
    text_io.new_line(24);
    Extract_Ops.Write_Buffer;
    Machine_Ops.Do_Command(Status,"$");

-- select some menu options for the buffer
elsif Buffer(1..2) = "SE" then
    Extract_Ops.Add_To_Extract_Buffer(
        Buffer      => Buffer,
        Menu_Used   => Menu_Used,
        Displayed   => Number_Of_Choices,
        Virtual_Path => Virtual_Path);
    Repeat_Screen := true;
    More := true;

-- help screen
elsif Buffer(1) = 'H' then
    Help_Screen;
    Repeat_Screen := true;
    More := true;

-- exit to vms
elsif Buffer(1..3) = "VMS" then
    Machine_Ops.Spawn(Status,"@geech$dua0:[stars_utilities]vms.com " &
        Virtual_Path.all);
    Repeat_Screen := true;

```

```

More := true;

-- goto first menu
elsif Buffer(1..4) = "MENU" then
    Repeat_Screen := false;
    More := false;
    Choice := -1;

-- review the users buffer of selected items
elsif Buffer(1) = 'R' then
    Extract_ops.Extract_Control;
    Repeat_Screen := true;
    More := true;
else
    -- get an integer from the users entry
    Choose_Option (Buffer,
                    Temp_Number_Of_Choices,
                    Choice,
                    Buffer_Status);
    if Buffer_Status = false then

        -- user entered invalid command (except for ' ')
        if Buffer(1) /= ' ' then
            text_io.new_line(12);
            text_io.put_line("'" & Buffer(1..length) & "'" &
                             " is not valid.");
            text_io.put_line("Type HELP for a list of valid commands.");
            text_io.new_line(12);
            delay(2.0);
            -- repeat this screen
            Repeat_Screen := true;
            More := true;
        else
            -- recursive call to get_menu_value
            Get_Menu_Value (Menu_Used,
                            Number_Of_Choices,
                            Longest_Menu_Line,
                            Title,
                            Virtual_Path,
                            Choice);

            -- found a valid command, so don't loop anymore
            More := false;
        end if;

    else
        -- found a valid command, so don't loop anymore
        More := false;
    end if;

end if;

-- reset the buffer for next command
Buffer(1..Buffer'last) := (others => ' ');

end loop; -- more loop

exception
when others =>
    null; -- Do not bomb out ... go gracefully.
end Get_Menu_Value;

end Menu_Ops;

```

with text_io;
with String Utilities;
with Condition_Handling;
with Lib;

Package Browser_Ops is

-- Description

-- This package contains the visible operations on the object Directory
-- Browser.

-- Requirements Satisfied

-- This package does not meet any specific STARS requirements.

-- Exceptions Raised

-- None.

-- Contact

-- Charles Elsner
-- Boeing Military Airplane
-- 3801 S. Oliver (ms K80-13)
-- Wichita, KS 67277

-- Implementation Dependencies and Assumptions:

-- Currently, this package uses some of the predefined types from package
-- Standard.

-- Revision History

-- Part Number:

-- Author: Charles Elsner

-- Version: 1.0;

-- Change Summary: Initial_Release

-- Date: 20-FEB-89

-- Distribution and Copyright

-- This Prologue must be included in all copies of this software.

-- This software is released to the Ada community.

-- This software is released to the Public Domain (Note: software released
-- to the Public Domain is not subject to copyright protection).

-- Restrictions on use or distribution: NONE

-- Disclaimer

-- This software and its documentation are provided "AS IS" and
-- without any expressed or implied warranties whatsoever.

-- No warranties as to performance, merchantability, or fitness
-- for a particular purpose exists.

-- Because of the diversity of conditions and hardware under
-- which this software may be used, no warranty of fitness for
-- a particular purpose is offered. The user is advised to test
-- the software thoroughly before relying on it. The user
-- must assume the entire risk and liability of using the software.

-- In no event shall any person or organization or people be
-- held responsible for any direct, indirect, consequential
-- or inconsequential damages or lost profits.

procedure Run_Browser(

```

Menu_1      : in string; --| VMS file name of the first menu
--| This file must be in top_level_dir
Menu_1_title : in string; --| Title to be displayed for menu_1
Menu_2      : in string; --| VMS file name of the second menu
--| This file must be in top_level_dir
Menu_2_Title : in string; --| Title to be displayed for menu_2
Editor_Used  : in string; --| Editor to use/plus options
Utility_Directory : in string; --| Locale of utility directory
Top_Level_Dir : in string; --| Directory to start browsing.
--| Menu_2 is a directory listing of
--| this_top_level_directory.

```

```

--| Description
--| This is the procedure to start the directory browser menu/interface
--| tool. Menu_1 is a string to identify the file name of the first menu.
--| This is an example of the menu_1 menu.

--| This is the first menu for the stars foundation
--| project search/view tool. ** Make certain that this file is version 1!
--| (Do not delete these first three lines. Program dependent.)
--| Review STARS Foundation Projects
--| Exit to VMS (in Directory ODIESDUAL28:[STARS_FOUNDATION])
--| Help
--| or 'X' to Exit (Return to SDME)
--| (Do not delete these last two lines. Program Dependent.)

--| Menu_2 is just a dir/col=1/output=Menu_2.dat.
--|
--| Exceptions Raised
--| None.

end Browser_Ops;

```

with Browser_Types;
with Calendar;
with Extract_Ops;
with File_Ops;
with Machine_Ops;
with Menu_Ops;
with String_Uutilities;
with Text_IO;

package body Browser_Ops is

-- | Description
-- | This package contains the internal operations on the object Directory
-- | Browser.
-- |
-- | Requirements Satisfied
-- | This package does not meet any specific STARS requirements.
-- |
-- | Exceptions Raised
-- |
-- | None.
-- |
-- | Contact
-- | Charles Elsner
-- | Boeing Military Airplane
-- | 3801 S. Oliver (ms K80-13)
-- | Wichita, KS 67277
-- |
-- | Implementation Dependencies and Assumptions:
-- | Currently, this package uses some of the predefined types from package
-- | Standard.
-- |
-- | Revision History
-- | Part Number:
-- | Author: Charles Elsner
-- | Version: 1.0;
-- | Change Summary: Initial_Release
-- | Date: 20-FEB-89
-- |
-- | Distribution and Copyright
-- | This Prologue must be included in all copies of this software.
-- | This software is released to the Ada community.
-- | This software is released to the Public Domain (Note: software released
-- | to the Public Domain is not subject to copyright protection).
-- | Restrictions on use or distribution: NONE
-- |
-- | Disclaimer
-- | This software and its documentation are provided "AS IS" and
-- | without any expressed or implied warranties whatsoever.
-- |
-- | No warranties as to performance, merchantability, or fitness
-- | for a particular purpose exists.
-- |
-- | Because of the diversity of conditions and hardware under
-- | which this software may be used, no warranty of fitness for
-- | a particular purpose is offered. The user is advised to test
-- | the software thoroughly before relying on it. The user
-- | must assume the entire risk and liability of using the software.
-- |
-- | In no event shall any person or organization or people be
-- | held responsible for any direct, indirect, consequential
-- | or inconsequential damages or lost profits.

--||

```
-----
procedure execute_choice (
    Choice          : in integer;
    User            : in out Browser_Types.String_Access_Type;
    Virtual_Path    : in out Browser_Types.String_Access_Type;
    Title           : in out Browser_Types.String_Access_Type;
    Utility_Directory : in out Browser_Types.String_Access_Type;
    Editor_Used     : in out Browser_Types.String_Access_Type;
    Ptr             : in out Browser_Types.Menu_List_Type;
    Go_Back         : in out Boolean) is

--| Description
--|   This procedure executes an operation based on the input Choice.  The main
--|   operations to be performed are, 1) edit a file, 2) Browse a Directory.
--|   To browse a directory, the procedure uses a recursive call.
--|

Current          : Browser_Types.Menu_List_Type := Ptr;
File_Type        : Browser_Types.String_Access_Type;
File_Name        : Browser_Types.String_Access_Type;
Command          : Browser_Types.String_Access_Type;
Menu_Head        : Browser_Types.Menu_List_Type;
Menu_Choices     : natural;
Menu_Longest     : natural;
New_Choice       : integer;
Status           : Browser_Types.Cond_Value_Type;
Data_File        : Text_IO.File_Type;

begin
-----
-- build Command string
-----
-- is choice a menu option
if Choice > 0 then

    -- find the actual line to execute (edit a file or browse a dir)
    for i in 1..(Choice - 1) loop
        Current := Current.Next;
    end loop;

    -- find the file type
    File_Type := new string'(String_Uutilities.File_Type_Is(Current.Line.all));

    -- is file a directory
    if File_Type.all = "DIR" then

-----
-- create, open and read data file, then delete it.
-----

        -- create a dir/col=1/output=file_name.dat
        File_Ops.Create_Data_File(
            Utility_Dir => Utility_Directory,
            File_Line   => Current.Line,
            User        => User,
            Virtual_Path => Virtual_Path,
            File         => File_Name);

        -- read the file into a linked list
        File_Ops.load_menu_list (
            File_Name      => File_Name.all,
            Ptr            => Menu_Head,
```

```

Data_File      => Data_File,
Number_Of_Choices => Menu_Choices,
longest_menu_line => Menu_Longest);

```

```

-- we are done with the file now
Text_Io.Delete(Data_File);

```

```

-----
-- display menu
-----

```

```

-- enter a loop for each Virtual_Path, displaying the choices
begin
Virtual: loop

```

```

-- display the menu and get a user command
Menu_Ops.Get_Menu_Value
  (Menu_Used      => Menu_Head,
   Number_Of_Choices => Menu_Choices,
   longest_menu_line => Menu_Longest,
   Title         => Virtual_Path,
   Virtual_Path   => Virtual_Path,
   Choice         => New_Choice);

```

```

-----
-- execute choice
-----

```

```

Execute_Choice (
  Choice      => New_Choice,
  User        => User,
  Virtual_Path => Virtual_Path,
  Title       => Current_line,
  Utility_Directory => Utility_Directory,
  Editor_Used  => Editor_Used,
  Ptr         => Menu_Head,
  Go_Back     => Go_Back);

```

```

-- exit loop when new_choice is 0
exit Virtual when New_Choice = 0;

```

```

-- if the user want to go back to main menu
exit Virtual when Go_Back;

```

```

end loop Virtual;
exception
  when others =>
    null;
end;

```

```

-- back out of the virtual path
Virtual_Path := new string'(String_Uilities.Back_Off(Virtual_Path.all));

```

```

else

```

```

-- edit the file
File_Ops.Edit_File(
  Virtual_Path => Virtual_Path,
  Editor       => Editor_Used,
  Line         => Current_line);

```

```

Go_Back := False;

```

```

end if;

```

```

elsif Choice = 0 then
  Go_Back := False;

```

```

elsif Choice < 0 then
    Go_Back := True;
else
    null; -- for future expansion
end if;

end Execute_Choice;

```

```

-----

procedure Execute_Main_Choice (
    Choice           : in out integer;
    Menu_2           : in string;
    Virtual_Path     : in out Browser_Types.String_Access_Type;
    Utility_Directory : in out Browser_Types.String_Access_Type;
    Editor_Used      : in out Browser_Types.String_Access_Type;
    User             : in out Browser_Types.String_Access_Type;
    Title            : in out Browser_Types.String_Access_Type) is

```

```

--| Description
--|   This routine will execute the users choice fro the main menu (ie menu_1)
--||

```

```

Command       : Browser_Types.String_Access_Type;
Menu_Head     : Browser_Types.Menu_List_Type;
Menu_Choices  : natural;
Menu_Longest  : natural;
New_Choice    : integer;
Status        : Browser_Types.Cond_Value_Type;
Data_File     : Text_IO.File_Type;
Go_Back       : Boolean := false;

```

```

begin

```

```

-- build Command string
-----

```

```

case Choice is

```

```

    when 0 =>
        null;

```

```

    -- browse a directory
    when 1 =>

```

```

        -- open and read from Ptr.line (ie the data file)
        -----

```

```

        File_Ops.load_menu_list (
            File_Name    => Menu_2,
            Ptr          => Menu_Head,
            Data_File    => Data_File,
            Number_Of_Choices => Menu_Choices,
            longest_menu_line => Menu_Longest);

```

```

        -----
        -- display menu
        -----

```

```

        -- enter a loop for first Virtual_Path, displaying the Choices

```

```

        begin
        First: loop

```

```

            Menu_Ops.Get_Menu_Value(
                Menu_Used      => Menu_Head,
                Number_Of_Choices => Menu_Choices,

```

```

longest_menu_line => Menu_Longest,
Title             => Title,
Virtual_Path      => Virtual_Path,
Choice            => New_Choice);

```

```

-----
-- execute Choice
-----

```

```

Execute_Choice (
    Choice           => New_Choice,
    User             => User,
    Virtual_Path     => Virtual_Path,
    Title            => Menu_Head_line,
    Utility_Directory => Utility_Directory,
    Editor_Used      => Editor_Used,
    Ptr              => Menu_Head,
    Go_Back          => Go_Back);

```

```

-- exit loop when New Choice is 0
exit first when New_Choice = 0;

```

```

-- user wants to go back to main menu
exit first when Go_Back;

```

```

end loop first;
exception
    when others =>
        null;
end;

```

```

when 2 =>
    Menu_Ops.Help_Screen;
when 3 =>
    Choice := 0;
when others =>
    null;
end case;

end Execute_Main_Choice;

```

```

-----
procedure Run_Browser(
    Menu_1           : in string;
    Menu_1_Title     : in string;
    Menu_2           : in string;
    Menu_2_Title     : in string;
    Editor_Used      : in string;
    Utility_Directory : in string;
    Top_Level_Dir    : in string) is

```

```

--| Description
--| Main call to execute the directory browser routine.
--|

```

```

Main_Head           : Browser_Types.Menu_List_Type;
Choice              : integer;
Number_Of_Choices   : natural := 0;
longest_menu_line   : natural := 0;
Virtual_Path        : Browser_Types.String_Access_Type :=
    new_string'(Top_Level_Dir);
Utility_Dir         : Browser_Types.String_Access_Type := new
    string'(Utility_Directory);

```

```

Editor          : Browser_Types.String_Access_Type := new
                  string'(Editor_Used);
Data_File       : Text_IO.File_Type;

Top_Level_Title : Browser_Types.String_Access_Type :=
  new_string'(Menu_1_Title);
Second_Level_Title : Browser_Types.String_Access_Type :=
  new_string'(Menu_2_Title);
Year            : calendar.year_number;
Month           : calendar.month_number;
Day             : calendar.day_number;
Seconds         : calendar.day_duration;
User            : Browser_Types.String_Access_Type;

begin

-----
-- Generate Unique user number for file contention prevention.
-----

    calendar.split(
        date      => calendar.clock,
        year      => Year,
        Month     => Month,
        Day       => Day,
        Seconds   => Seconds);

    -- use the seconds to create the users unique number
    User := new_string'(String_utilities.Remove_Leading_And_Trailing_Blanks(
        integer'image(integer(Seconds))));

-----
-- turn off system messages
-----

--    Machine_Ops.Turn_Off_Mess;

-----
-- open and read from Menu_1.dat
-----

    File_Ops.Load_Menu_List (
        File_Name      => Menu_1,
        Ptr            => Main_Head,
        Data_File      => Data_File,
        Number_Of_Choices => Number_Of_Choices,
        longest_menu_line => longest_menu_line);

-----
-- display menu
-----

    loop

        Menu_Ops.Get_Menu_Value(
            Menu_Used      => Main_Head,
            Number_Of_Choices => Number_Of_Choices,
            longest_menu_line => longest_menu_line,
            Title          => Top_Level_Title,
            Virtual_Path   => Virtual_Path,
            Choice         => Choice);

-----
-- execute main Choice

```

Execute_Main_Choice (

Choice	=> Choice,
Menu_2	=> Menu_2,
Virtual_Path	=> Virtual_Path,
Utility_Directory	=> Utility_Dir,
Editor_Used	=> Editor,
User	=> User,
Title	=> Second_Level_Title);

```
if Choice = 0 then
  exit;
end if;
```

```
end loop;
```

-- Write the Extraction Buffer to the user Directory.

```
Extract_Ops.Write_Buffer;
```

-- final greeting

```
text_io.new_line(24);
```

```
-- Machine_Ops.Turn_On_Mess;
```

```
exception
when others =>
  null; -- go gracefully
```

```
end Run_Browser;
```

```
end Browser_Ops;
```

with Browser_Ops;
with Browser_Types;

procedure Browser_Driver is

Description

This procedure calls the Run_Browser procedure.

Requirements Satisfied

This package does not meet any specific STARS requirements.

Exceptions Raised

None.

Contact

Charles Elsner
Boeing Military Airplane
3801 S. Oliver (ms K80-13)
Wichita, KS 67277

Implementation Dependencies and Assumptions:

Currently, this package uses some of the predefined types from package Standard.

Revision History

Part Number:
Author: Charles Elsner
Version: 1.0;
Change Summary: Initial_Release
Date: 20-FEB-89

Distribution and Copyright

This Prologue must be included in all copies of this software.
This software is released to the Ada community.
This software is released to the Public Domain (Note: software released to the Public Domain is not subject to copyright protection).
Restrictions on use or distribution: NONE

Disclaimer

This software and its documentation are provided "AS IS" and without any expressed or implied warranties whatsoever.

No warranties as to performance, merchantability, or fitness for a particular purpose exists.

Because of the diversity of conditions and hardware under which this software may be used, no warranty of fitness for a particular purpose is offered. The user is advised to test the software thoroughly before relying on it. The user must assume the entire risk and liability of using the software.

In no event shall any person or organization or people be held responsible for any direct, indirect, consequential or inconsequential damages or lost profits.

-- Top level directory that the browser will start at

Top_Level_Dir : Browser_Types.String_Access_Type := new
string'("ODIESDUAL28:[STARS_FOUNDATION]");

-- Title that will be displayed for menu 1

Menu_1_Title : Browser_Types.String_Access_Type := new

```
string'("STARS Foundation");
```

```
-- Path name for first menu
```

```
Menu_1 : Browser_Types.String_Access_Type := new  
    string'( "geeCh$dUA0:[stars_utilities]found_menu_1.dat;1");
```

```
-- Title that will be displayed for menu 2
```

```
Menu_2_Title : Browser_Types.String_Access_Type := new  
    string'("STARS Foundation Projects");
```

```
-- Path name for second menu
```

```
Menu_2 : Browser_Types.String_Access_Type := new  
    string'( Top_Level_Dir.all & "menu_2.dat;1");
```

```
-- editor that the users will use
```

```
Editor : Browser_Types.String_Access_Type := new string'("edit/tpu");
```

```
-- utility directory where the file operations will take place
```

```
Utility_Directory : Browser_Types.String_Access_Type := new  
    string'("GEECH$dUA0:[STARS_UTILITIES]");
```

```
begin
```

```
-- call the procedure to start the browser
```

```
Browser_Ops.Run_Browser(  
    Menu_1           => Menu_1.all,  
    Menu_1_Title     => Menu_1_Title.all,  
    Menu_2           => Menu_2.all,  
    Menu_2_Title     => Menu_2_Title.all,  
    Editor_Used      => Editor.all,  
    Utility_Directory => Utility_Directory.all,  
    Top_Level_Dir    => Top_Level_Dir.all);
```

```
end;
```



```

$! This is the command file used in the 'escape to vms' function in the
$! directory browser. The file name VMS.COM. '!' is a comment token.
$!
$ ESC[0,8]=27
$ WS      := WRITE SYS$OUTPUT      !write to screen
$ C_HOME  := 'ESC'[H              !go to upper left part of screen
$ CLEAR   := 'ESC'[2J             !clear screen
$ set nocontrol=y
$ save_dir = f$environment("default")
$ set def 'p1'                    !p1 is the input parameter (dir)
$ WS C_HOME,CLEAR
$ WS "*****"
$ WS " *          Type 'X' or 'EXIT' to return to the Browser          *"
$ WS " *          Type 'HELP' to invoke VAX/VMS help facility          *"
$ WS " *          Type 'HELP_B' to invoke Browser help facility        *"
$ WS " *          Type 'SYMBOLS' to see the VAX/VMS Symbols set up for you  *"
$ WS "*****"
$ WS " "
$ WS " Current Directory: "
$ sho def
$ WS " "
$!
$ ready:
$   inquire,nopunctuation next "browser_$ "
$   if next .nes. "HELP_B" then goto not_help
$   type geech$dual0:[stars_utilities]vms_help.txt
$   goto ready
$ not_help:
$   if next .eqs. "KERMIT" then next = "RKERMIT"
$   if next .eqs. "DONE" then goto isdone
$   if next .eqs. "EXIT" then goto isdone
$   if next .eqs. "QUIT" then goto isdone
$   if next .eqs. "STOP" then goto isdone
$   if next .eqs. "LO" then goto isdone
$   if next .eqs. "LOG" then goto isdone
$   if next .eqs. "LOGOUT" then goto isdone
$   if next .eqs. "LOGOFF" then goto isdone
$   if next .eqs. "MENU" then goto isdone
$   if next .eqs. "BROWSER" then goto isdone
$   if next .eqs. "HOME" then goto home_done
$   if next .eqs. "X" then goto isdone
$!
$   define/user_mode/NOLOG sys$input sys$command
$   set control=y
$   on control_y then continue
$   on severe_error then continue
$   'next'                    !execute the users vms command
$   set nocontrol=y
$   goto ready
$ home_done:
$   set def sys$login
$   goto ready
$ isdone:
$   set control=y
$   set def 'save_dir'      !go back to users original directory
$   WS C_HOME,CLEAR
$   exit                    ! exit this command file

```