```
// ##########################################################################################
// ##########################################################################################
//
// libFilesys.mel
// ------------------------------------------------------------------------------------------
// Implements platform independant and 'safe' wrappers around file system and I/O
// functions.
// NOTE: THIS IS AN INCOMPLETE LISTING FOR SAMPLE PURPOSES ONLY
//
// AUTHOR:  Nick Gray
// CREATED: 2003-03-03
// REVISED: 2003-04-14 ... Extracted these functions from deprecated "Globals" module

// FUNCTIONS - All functions in the module are prefixed with "libFilesys_"
//             Descriptions prefixed with '***' are functions that have not been written yet
//
// Version()                            - Current version of this library
//
// GENERAL PLATFORM INDEPENDANT FILESYSTEM FUNCTIONS
// GetScriptPath()                      - Returns the absolute path of the location of the libTex tools
// GetIconPath(iconName)                - Returns the absolute path to a specified libTex icon
// GetUserName()                        - Returns the name of the current Maya user
// GetUserPrefsDir()                    - Returns directory in which user prefs are stored - usually same as maya's user prefs dir
// OpenHelpFile(pageName)               - Opens the specified libTex help page
// SaveSetDefinition(path, list)        - Saves the specified sets to a disk file
// LoadSetDefinition(path, mode)        - Loads set information from a file saved with "SaveSetDefinition"

// ------------------------------------------------------------------------------------------
// INCLUDES
libTexPaintLib_Include("libTexPaintLib");

// ------------------------------------------------------------------------------------------
// GLOBAL VARS
global string $libFilesys_ScriptDir_DoNotReferenceDirectly = "";       // Global string contains current directory
                                                                       // use libFilesys_GetScriptPath() to access

// ##########################################################################################
// VERSION
// ##########################################################################################
global proc string libFilesys_Version()
{  return("libFilesys 1.0");
}

// ##########################################################################################
// PLATFORM INDEPENDANT PATH AND FILE HANDLING
// ##########################################################################################

// ------------------------------------------------------------------------------------------
/*
       Name:          GetScriptPath
       Fcn Type:      Filesystem
       Args:          None
```

```
        Returns:        String
        Success:        Path to libTexPaint root directory
        Error:          Empty string

        Returns the path of the directory where these scripts are located. Note that functions
        should always call this function to get libTexPaint paths and not rely on any other method.

        History:        2003-04-14:    Created
        Related:        libFilesys_GetIconPath
*/
// ---------------------------------------------------------------------------
global proc string libFilesys_GetScriptPath()
{
   global string $libFilesys_ScriptDir_DoNotReferenceDirectly;         // Global sring contains current directory

   if ("" == $libFilesys_ScriptDir_DoNotReferenceDirectly)             // If the global has not been set yet...
   {  string $sPath = `whatIs libFilesys_GetScriptPath`;               // ... derive it once only by finding the location of this file
      string $sArray[];                                                        // Make an array for tokenizing...
      tokenize $sPath " " $sArray;                                             // ... and split the path out of the "whatIs" message
      $libFilesys_ScriptDir_DoNotReferenceDirectly = `substring $sArray[4] 1 (size($sArray[4])-size("libTexPaintFileSysLib.mel"))`;
   }
   return($libFilesys_ScriptDir_DoNotReferenceDirectly);               // Return the path
}

// ---------------------------------------------------------------------------
/*
        Name:           GetIconPath
        Fcn Type:       Filesystem
        Args:           None
        Returns:        String
        Success:        Path to libTexPaint icons directory
        Error:          Empty string

        Returns a path to the [.icons] directory where images for libTexPaint tools are stored.
        Always call this function to get icon paths.

        History:        2003-04-14:    Created
        Related:        libFilesys_GetScriptPath
*/
// ---------------------------------------------------------------------------
global proc string libFilesys_GetIconPath(string $sArgIcon)
{  return(libFilesys_GetScriptPath() + ".icons/" + $sArgIcon);         // Simple wrapper lets us change the location of the icon dir
}

// ---------------------------------------------------------------------------
/*
        Name:           GetUserName
        Fcn Type:       Filesystem
        Args:           None
        Returns:        String
        Success:        Name of current user
        Error:          Empty string
```

```
        Returns the user name of the current Maya user (not the current user project name).

        History:        2003-04-14:    Created
        Related:        libFilesys_GetUserPrefsDir
*/
// ---------------------------------------------------------------------------
global proc string libFilesys_GetUserName()
{
   string $sUser = getenv("USER");                                          // Extract from environment var
   if ("" == $sUser)                                                        // ... check to see if we got a name
   {  $sUser = getenv("user");                                              // ... try using lowercase, just in case
   }
   return($sUser);                                                          // Return the found name
}

// ---------------------------------------------------------------------------
/*
        Name:           GetUserPrefsDir
        Fcn Type:       Filesystem
        Args:           None
        Returns:        String
        Success:        Current user prefs directory
        Error:          Empty string

        Returns a path to the users preferences directory - this is O/S dependant, since
        the users home dir is different on *Nix and NT systems.

        History:        2003-04-14:    Created
        Related:        libFilesys_GetUserName
*/
// ---------------------------------------------------------------------------
global proc string libFilesys_GetUserPrefsDir()
{
   string $sPath = `about -env`;                                            // Get the location of the maya en
   string $sFile = basename($sPath, "");                                    // Get the env file name
   return(substring($sPath, 1, size($sPath) - size($sFile)));               // Trim off the trailing file name
}

// ---------------------------------------------------------------------------
/*
        Name:           OpenHelpFile
        Fcn Type:       Filesystem
        Args:           String - name of help file
        Returns:        None

        Displays help files located in the libTexPaint [.help] directory using Mayas help subsystem.

        History:        2003-04-14:    Created
        Related:        libFilesys_ GetScriptPath
*/
// ---------------------------------------------------------------------------
```

```
global proc libFilesys_OpenHelpFile(string $sArgPage)
{  showHelp -a (libFilesys_GetScriptPath() + ".help/" + $sArgPage);    // Simple wrapper lets us change the location of the help dir
}

// ---------------------------------------------------------------------------
/*
       Name:          SaveSetDefinition
       Fcn Type:      data export
       Args:          String - full path to output file
                      stringArray - list of sets to export
       Returns:       Int
       Success:       TRUE if file is written successfully
       Error:         FALSE if output failed

       Writes a file containing set definitions for provided set list. The setdef file
       the contains the minimum amount of information to fully reconstruct both object and
       shader sets. The format is as follows:
              HEADER   - Always <libTex_setdef>
              SET TYPE - <geo> or <shd> - either geomemetry or shader set
              SET NAME - SETNAME - plain-text set name
              SHD NAME - SHADERNAME - Only appears in <shd> sets for linked shader name
              SURFACES - MAYA-SURFACE-PATH  - The long name of each surface contained in set
                 ...                        - List continues until a tag line is reached
              END      - <eof> - End of definition file
       The file is read until the EOF tag or an actual EOF occurs, so the EOF tag is not
       strictly necessary. Note that no information other than membership is stored.
       Returns TRUE if a file was successfully saved.
       Note that currently only writes per-object and per-facet shading.

       History:       2003-04-14:    Created
       Related:       libFilesys_ LoadSetDefinition
*/
// ---------------------------------------------------------------------------
global proc int libFilesys_SaveSetDefinition(string $sArgPath, string $sArgSetList[])
{
   string $sSetList[] = libTexPaintLibSet_GetSetListFromSetList($sArgSetList);    // Sanitize the set list
   if (("" == $sArgPath) || (1 > size($sSetList)))          { return(false); }   // Insufficient arg info - exit

   string $sFile = basename($sArgPath, ".set") + ".set";                         // Make a proper name
   if (5 > size($sFile))                                    { return(false); }   // No file name = nothing saved
   string $sPath = dirname($sArgPath) + "/";                                     // Clip the path

   int    $iFileHandle;                                                          // Handle to output file
   string $sSet;                                                                 // Var for walking set list
   string $sNodeList[];                                                          // List of nodes in set
   string $sNode;                                                                // Var for walking node list
   string $sRenderNode;                                                          // Var holds name of set if it is a shader

   $iFileHandle = fopen(($sPath + $sFile), "w");                                 // Open file for writing
   if (!$iFileHandle)                                                            // If no handle was returned...
   {  warning ("Could not open file " + $sPath + $sFile + " for writing.");      // ... warn
      return(false);                                                             // ... and exit
```

```
   }

   fprint($iFileHandle, "<libTex_setdef>\n");                                  // Write header

   for ($sSet in $sSetList)                                                    // Walk the set list
   {  $sRenderNode = libTexPaintLibSet_GetShaderNodeFromSet($sSet);            // Determine if it's a shading set
      if ("" == $sRenderNode)                                                  // If it's not a shading set...
      {  fprint($iFileHandle, "<geo>\n");                                      // ... write geometry header
         fprint($iFileHandle, ("  " + $sSet + "\n"));                         // ... write the set name
      }
      else
      {  fprint($iFileHandle, "<shd>\n");                                      // Otherwise, it is a shader...
         fprint($iFileHandle, ("  " + $sSet + "\n"));                         // ... write the shader set name
         fprint($iFileHandle, ("  " + $sRenderNode + "\n"));                  // ... write the shader node name
         if (libTexPaintLibTexture_GetPerSurfaceTexAttrib($sSet))             // ... if it's a global shader...
         {  fprint($iFileHandle, ("  global\n"));                            // ... write global identifier
         }
         else                                                                 // ... otherwise...
         {  fprint($iFileHandle, ("  local\n"));                             // ... write local identifier
         }
      }
      $sNodeList = libTexPaintLibSet_GetNodeListFromSet($sSet);              // Get the constituant nodes
      for ($sNode in $sNodeList)                                             // Walk the nodes in the set...
      {  fprint($iFileHandle, ("    " + $sNode + "\n"));                     // ... write the node name
      }
   }
   fprint($iFileHandle, ("<eof>\n"));                                        // Write the EOF
   fclose $iFileHandle;                                                      // Close the file
   return(true);                                                             // All done
}

// ----------------------------------------------------------------------------
/*
       Name:          LoadSetDefinition
       Fcn Type:      data export
       Args:          String - full path to input file
                      Int - file import mode
       Returns:       Int
       Success:       TRUE if file is read successfully
       Error:         FALSE if read failed

       Loads a set definition file and connects the set to surface objects. If an existing
       set is found with the same name, then the nodes listed are forced into membership
       exclusively with that set. If no set exists, then a set is created. If a shading
       set must be created, then a shader is built and named accordingly.
       Mode arg determines if only render or object sets, or both, should be loaded…
              0 - Load All Sets
              1 - Load Rendering Sets Only
              2 - Load Object Sets Only

       History:       2003-04-14:    Created
       Related:       libFilesys_SaveSetDefinition
```

```
*/
// ----------------------------------------------------------------------------
global proc int libFilesys_LoadSetDefinition(string $sArgPath, int $iArgMode)
{
   if ("" == $sArgPath)                                    { return(false); }   // No path = no load
   if ((0 > $iArgMode) || (2 < $iArgMode))                 { $iArgMode = 0; }   // Sanitize the mode setting

   string $sFile = basename($sArgPath, ".set") + ".set";                        // Make a proper name
   if (5 > size($sFile))                                   { return(false); }   // No file name = nothing to load
   string $sPath = dirname($sArgPath) + "/";                                    // Clip the path
   if (!(`filetest -r ($sPath + $sFile)`))                 { return(false); }   // If the file does not exist... exit

   int $iFileHandle = fopen(($sPath + $sFile), "r");                            // Open file for reading
   if ("<libTex_setdef>\n" != `fgetline $iFileHandle`)                           // If the header isn't correct...
   {  fclose $iFileHandle;                                                      // Close the file
      return(false);                                                            // No set data read = bad load
   }

   string $sLine;                                                               // Var used for loading data
   string $sMsg;                                                                // Node name stored for warning messages
   string $sSet = "";                                                           // Set to add nodes to
   string $sNode;                                                               // Current node being added

   while (!(`feof $iFileHandle`))                                               // Loop until no lines are left
   {  $sLine = `fgetline $iFileHandle`;                                         // Get the next line in the file
      $sLine = strip(substring($sLine, 1, (size($sLine)-1)));                   // Trim newline char and remove whitespace
      if ("" == $sLine)                                                         // If we get a blank line...
      {  //print("Blank line ignored...\n");                                     // ... ignore it
      }
      else if ("#" == substring($sLine, 1, 2))                                  // If we get a comment line...
      {  //print("Comment line ignored...\n");                                  // ... ignore it
      }
      else if ("<geo>" == $sLine)                                               // If we get a geometry set...
      {  $sLine = `fgetline $iFileHandle`;                                      // Get the next line in the file
         $sLine = strip(substring($sLine, 1, (size($sLine)-1)));                // Trim newline char and remove whitespace
         $sSet  = $sLine;                                                       // Lock to the new current set
         if ($iArgMode == 1)                                                    // If the current mode turns off object sets...
         {  $sSet = "||OFF||";                                                  // ... disable accumulation
         }
         else
         {  if (1 > size(`ls -set $sSet`))                                      // If the set does not exist...
            {  $sSet = libTexPaintLibSet_CreateSelectSet($sSet);                // ... create it
            }
         }
      }
      else if ("<shd>" == $sLine)                                               // If we get a shading node...
      {  $sLine = `fgetline $iFileHandle`;                                      // Get the next line in the file
         $sLine = strip(substring($sLine, 1, (size($sLine)-1)));                // Trim newline char and remove whitespace
         $sSet  = $sLine;                                                       // Lock to the new current set
         $sLine = `fgetline $iFileHandle`;                                      // Get the next line in the file
         $sLine = strip(substring($sLine, 1, (size($sLine)-1)));                // Trim newline char and remove whitespace
         if ($iArgMode == 2)                                                    // If the current mode turns off rendering sets...
```

```
         {  $sSet = "||OFF||";                                                  // ... disable accumulation
         }
         else
         {  if (1 > size(`ls -set $sSet`))                                      // If the set does not exist...
            {  $sSet = libTexPaintLibSet_CreateRenderSet($sSet);                 // ... create render set - shader name is enfoced
            }
            $sLine = `fgetline $iFileHandle`;                                   // Get the next line in the file
            $sLine = strip(substring($sLine, 1, (size($sLine)-1)));             // Trim newline char and remove whitespace
            if ("global" == $sLine)                                             // If the shader is global...
            {  libTexPaintLibTexture_SetPerSurfaceTexAttrib($sSet, true);        // ... set its attrib to global
            }
         }
      }
      else if ("<eof>" == $sLine)                                               // If we get an end-of-file line...
      {  break;                                                                 // ... exit the scan loop
      }
      else
      {  if ("" != $sSet)                                                       // Make sure we have a set to add to
         {  $sMsg = $sLine;                                                     // Keep a copy of the node name for warnings
            while (true)                                                        // Loop until we get a result
            {  if ("" == $sLine)                                                // If the node name is dead...
               {  warning ("Could not reconnect node: " + $sMsg);               // ... spit out a warning
                  break;                                                        // ... give up
               }
               $sNode = libTexPaintLibNode_GetNode($sLine);                     // Check to see if the name matches a scene element
               if ("" != $sNode)                                                // If we found a match...
               {  if ("||OFF||" != $sSet)                                       // ... if set accumulation is not disabled...
                  {  sets -fe $sSet $sNode;                                     // ... include the object in the set
                  }
                  break;                                                        // ... exit while loop
               }
               $sLine = libTexPaintLibNode_GetHierarchyNameRemoveHead($sLine);  // Still here? Try to match shorter ver of name
            }
         }
      }
   }
   fclose $iFileHandle;                                                         // Close the file
   return(true);                                                                // All done
}
```