The ENVIRONMENT MANAGER

December 7, 1984

2600 Liberty Avenue
P. O. Box 2600
Pittsburgh, PA 15230
(412) 355-0900

Accent is a trademark of Carnegie-Mellon University.

Accent and many of its subsystems and support programs were originally developed by the CMU Computer Science Department as part of its Spice Project.

This document is adapted from the paper by Mary R. Thompson and Michael B. Jones, *Preliminary Environment Manager.* Carnegie-Mellon University, Pittsburgh, PA, 1984.

The information in this document is subject to change without notice and should not be construed as a commitment by PERQ Systems Corporation. The company assumes no responsibility for any errors that may appear in this document.

PERQ Systems Corporation will make every effort to keep customers apprised of all documentation changes as quickly as possible. The Reader's Comments card is distributed with this document to request users' critical evaluation to assist us in preparing future documentation.

PERQ, PERQ2, LINQ, and Qnix are trademarks of PERQ Systems Corporation.

Table of Contents Page

1. Theory EN-1
- 1.1. Variable Types and Values EN-1
- 1.2. Variable Scopes EN-2
- 1.3. Recursive Searchlist Calling EN-2

2. Definitions EN-5
- 2.1. Type Definitions EN-5
- 2.2. Routine Definitions EN-7
  - 2.2.1. Getting Environment Manager version number EN-7
  - 2.2.2. Getting an environment variable value EN-7
  - 2.2.3. Entering a new variable EN-8
  - 2.2.4. Resolving searchlist variable value EN-9
  - 2.2.5. Listing variables by name EN-10
  - 2.2.6. Copying an environment manager connection EN-11
  - 2.2.7. Destroying environment manager connection EN-12

1. Theory

The Environment Manager provides a language- and process-independent message interface for defining and retrieving environment variables. This allows different processes on the same machine to share directory searchlists and other variables.

Each process ordinarily has its own connection to the Environment Manager. This is set up by the parent of that process at the moment of creation and is available in the variable EMPort, defined in PascalInit.Pas in LibPascal. The message interface for a process is initialized by process startup code; the InitEnvMgr routine (found in EnvMgrUser.Pas in LibPascal) need not be called.

An environment variable is a named variable that has the following attributes:

- Type
- Scope
- Set of values

1.1. Variable Types and Values

The type of an environment variable can be either string or searchlist. A string-type variable can have any arbitrary string as its value.

A searchlist-type variable holds a list of directories to be searched when looking for a file. The values of a searchlist variable can be either directory names or names of other searchlists. When the searchlist variable is used, all references to other searchlists are

expanded (replaced by their contents) until the expanded searchlist consists only of directory names. Searchlist value entries that are references to other searchlists are followed by a colon (":") and optionally a subdirectory name. This is the same syntax that is used by the file system to denote searchlists.

At the shell level, searchlist-type environment variable names are distinguished from string-type variable names by having a colon as the last character of the name. This colon is NOT used by any of the Environment Manager interface routines and is not returned by the ScanEnvVariables routine (explained in Section 2.2).

1.2. Variable Scopes

An environment variable can be either local or global in scope. A local variable is seen only by a single process, while global variables are visible to all the processes that are served by the Environment Manager.

When a child process is created from a parent process, the local environment (the environment variables defined as being local to the parent process) of the parent is copied to the child. However, any subsequent changes made in the parent's local environment are not shared with the child process.

1.3. Recursive Searchlist Calling

Ordinarily, if a local searchlist exists with the same name as a global searchlist, the local searchlist will be used. However, if the local searchlist contains a reference to itself, this reference is interpreted as a reference to a global searchlist of the same name, not as a recursive reference to the local searchlist. This is useful because it allows the system to specify a default searchlist for a given subsystem by name and for that subsystem to reference the searchlist by that name, while at the same time allowing the user to define a local searchlist with the same name to override the

default searchlist. The user can then reference the default searchlist from within the local definition, allowing the user to add directories to the default searchlist.

A global searchlist can contain references to local as well as global searchlists. For example, Accent normally searches for run files within the "Run:" searchlist and all other files within the "Default:" searchlist. A user can include in the global "Run:" searchlist a reference to "Default:". Each time the "Run:" searchlist is resolved, the system will also search the "Default:" searchlist. Since the "Default:" searchlist is first resolved locally, this allows the one global definition of the "Run:" searchlist to refer to different "Default:" searchlists, depending on which process resolves the variable.

2. Definitions

The definitions throughout this document are given in Pascal. If you are programming in the C language, please refer also to the document "C System Interfaces" in the *Accent Languages Manual*. If you are programming in the Lisp language, see the document "Lisp Interaction with the Accent Operating System" in the *Accent Lisp Manual*. When FORTRAN becomes available under Accent, the definitions will be the same as in the C language.

2.1. Type Definitions

These type and constant definitions are found in module EnvMgrDefs in EnvMgrDefs.Pas in LibPascal.

```
{}
{ Env_Variable:  A list of environment entries, each of which
{        is a string.
{}
const
    Env_Element_Size    = 255;  { MaxString }
type
    Env_Element         = string[Env_Element_Size];
    Env_Element_Array   = array [0 .. 0] of Env_Element;
                                            { hack }
    Env_Variable        = ^ Env_Element_Array;

{}
{ A Searchlist name embedded in a searchlist string is followed
{ by a Searchlist_Separator character.
{}

const
    Searchlist_Separator  = ':';


{}
{ Env_Var_Name:  The name string for an environment variable.
```

```
{}
const
    Env_VarName_Size    = Entry_Name_Size;
type
    Env_Var_Name        = string[Env_VarName_Size];


{}
{ Env_Var_Type:  The environment variable type values.
{}
type
    Env_Var_Type = (
        Env_String,         { Values are lists of strings }
        Env_SearchList);    { Value is a searchlist }

{}
{ Env_Var_Scope:  Flag specifying whether to find environment
{       variable in the local table, global table, or using
{       the normal method of local and then global.
{}
type
    Env_Var_Scope = (
        Env_Normal,      { Use the normal lookup method }
        Env_Local,       { Refer to name in per-process table}
        Env_Global);     { Refer to name in global environment
                             variable table }


{}
{ Env_Scan_List:  A list of environment variable names, types,
{ and scopes.
{}

type
    Env_Scan_Record =
        record
        VarName          : Env_Var_Name;
        VarType          : Env_Var_Type;
        VarScope         : Env_Var_Scope;
        end;

    Env_Scan_Array       = array [0 .. 0] of
                                Env_Scan_Record;

    Env_Scan_List        = ^ Env_Scan_Array;
                               {variable length array}

{}
{ Error return values for Environment Manager.
{}

const
    Env_Error_Base       = 1600;

    EnvVariableNotFound = Env_Error_Base + 1;
    WrongEnvVarType      = Env_Error_Base + 2;
```

```
BadSearchlistSyntax = Env_Error_Base + 3;
SearchlistLoop      = Env_Error_Base + 4;
FirstItemNotDefined = Env_Error_Base + 5;
```

2.2. Routine Definitions

These routine definitions are in module EnvMgr in EnvMgrUser.Pas in LibPascal.

2.2.1. Getting Environment Manager version number

```
Function EnvMgr_Version(
                ServPort : Port;
            var Versn    : string
        ): GeneralReturn;
```

Abstract:

Returns version number of the Environment Manager.

Parameters:

ServPort Connection to Environment Manager

Versn Will be set to version number

2.2.2. Getting an environment variable value

```
function  GetEnvVariable(
                ServPort       : port;
                Name           : Env_Var_Name;
                SearchScope    : Env_Var_Scope;
            var Variable       : Env_Variable;
            var Variable_Cnt   : long;
            var VarType        : Env_Var_Type;
            var ActualScope    : Env_Var_Scope
         ):GeneralReturn;
```

Abstract:

Returns the value of an environment variable. If the variable is a searchlist, this does not evaluate any contained searchlist references.

Parameters:

ServPort Connection to Environment Manager for process

Name Name of environment variable

SearchScope
  Where to search:

Env_GlobalGlobal environment only

Env_Local
Local environment only

Env_Normal
Search the local environment. If the variable is not there, search the global environment.

Variable Returns a pointer to the variable (a variable-length array of strings)

Variable_Cnt
Returns the number of entries in variable

VarType Returns type of environment variable: Env_Searchlist or Env_String

ActualScope
Returns where the variable was actually found (Env_Local or Env_Global)

Returns:

Success

EnvVariableNotFound

2.2.3. Entering a new variable

```
function  SetEnvVariable(
              ServPort     : port;
              Name         : Env_Var_Name;
              VarType      : Env_Var_Type;
              VarScope     : Env_Var_Scope;
              Variable     : Env_Variable;
              Variable_Cnt : long
      ):GeneralReturn;
```

Abstract:

Enters a new environment variable. If the variable is a search list, checks for valid search list syntax and ensures that each entry ends in a directory separator ("/") or a searchlist terminator (":").

Parameters:

ServPort Connection to Environment Manager for process

Name Name of environment variable

VarType Type of variable to enter:

Env_String or Env_Searchlist

VarScope Where to enter the variable:
Env_Global global environment
Env_Local local environment

Variable Pointer to the variable (a variable-length array of strings)

Variable_Cnt
Number of elements in variable. If the value is zero (an empty array), the name is deleted.

Returns:

Success

BadName If search list name is null or if an entry is malformed. Null search list names are ignored.

2.2.4. Resolving searchlist variable value

```
function  ResolveSearchList(
              ServPort       : port;
              Name           : Env_Var_Name;
              FirstOnly      : boolean;
          var Variable       : Env_Variable;
          var Variable_Cnt   : long;
          var FirstDefined   : boolean
      ):GeneralReturn;
```

Abstract:

The ResolveSearchList call is used to resolve the value of an environment variable of type Env_SearchList, recursively expanding any environment variable references contained therein. If undefined names are encountered during the expansion, they are ignored and the expansion is continued. It is an error if the evaluation results in an empty search list.

Parameters:

ServPort Port for process environment

Name Name of search list

FirstOnly One of:

True Only return the first item in the expansion

False Return all items in the expansion

Variable Returns a pointer to the search list (a variable-length array of directory names)

Variable_Cnt
Returns number of entries in search list

FirstDefined
Returns one of:

True	If the first element in the expansion exists
False	If it could not be resolved (it was a reference to a search list that did not exist)

Returns:

Success

EnvVariableNotFound
WrongEnvVarType
SearchListLoop

2.2.5. Listing variables by name

```
function  ScanEnvVariables(
              ServPort          : port;
              SearchScope       : Env_Var_Scope;
          var EnvScanList       : Env_Scan_List;
          var EnvScanList_Cnt   : long
      ):GeneralReturn;
```

Abstract:

Lists the defined environment variables by name.

Parameters:

ServPort Port for process environment

SearchScope
One of the following:

Env_Globallist global variables only

Env_Local
list local variables only

Env_Normal
list all local variables, and all global variables that are

not hidden by local variables with the same names

EnvScanList
: Returns the list of variable names, types, and scopes

EnvScanList_Cnt
: Returns the number of entries in EnvScanList

Returns:

Success

2.2.6. Copying an environment manager connection

```
function  CopyEnvConnection(
              ServPort       : port;
              OldConnection  : port;
          var NewConnection  : port
      ):GeneralReturn;
```

Abstract:

Creates a new connection to the Environment manager, copying all of the local variables belonging to the old connection.

Parameters:

ServPort
: Any port to the Environment Manager

OldConnection
: Port designating parent connection. If it is NullPort, the new connection will have no local variables; otherwise, it will receive copies of all the local variables from OldConnection

NewConnection
: Returns port for new connection

Returns:

Success

Failure
: No more connections available

2.2.7. Destroying environment manager connection

```
function EnvDisConnect(
                ServPort : port
        ): GeneralReturn;
```

Abstract:

Destroys a connection to the Environment manager and all associated variables.

Parameters:

ServPort Port designating connection. The port will be deallocated.

Returns:

Success