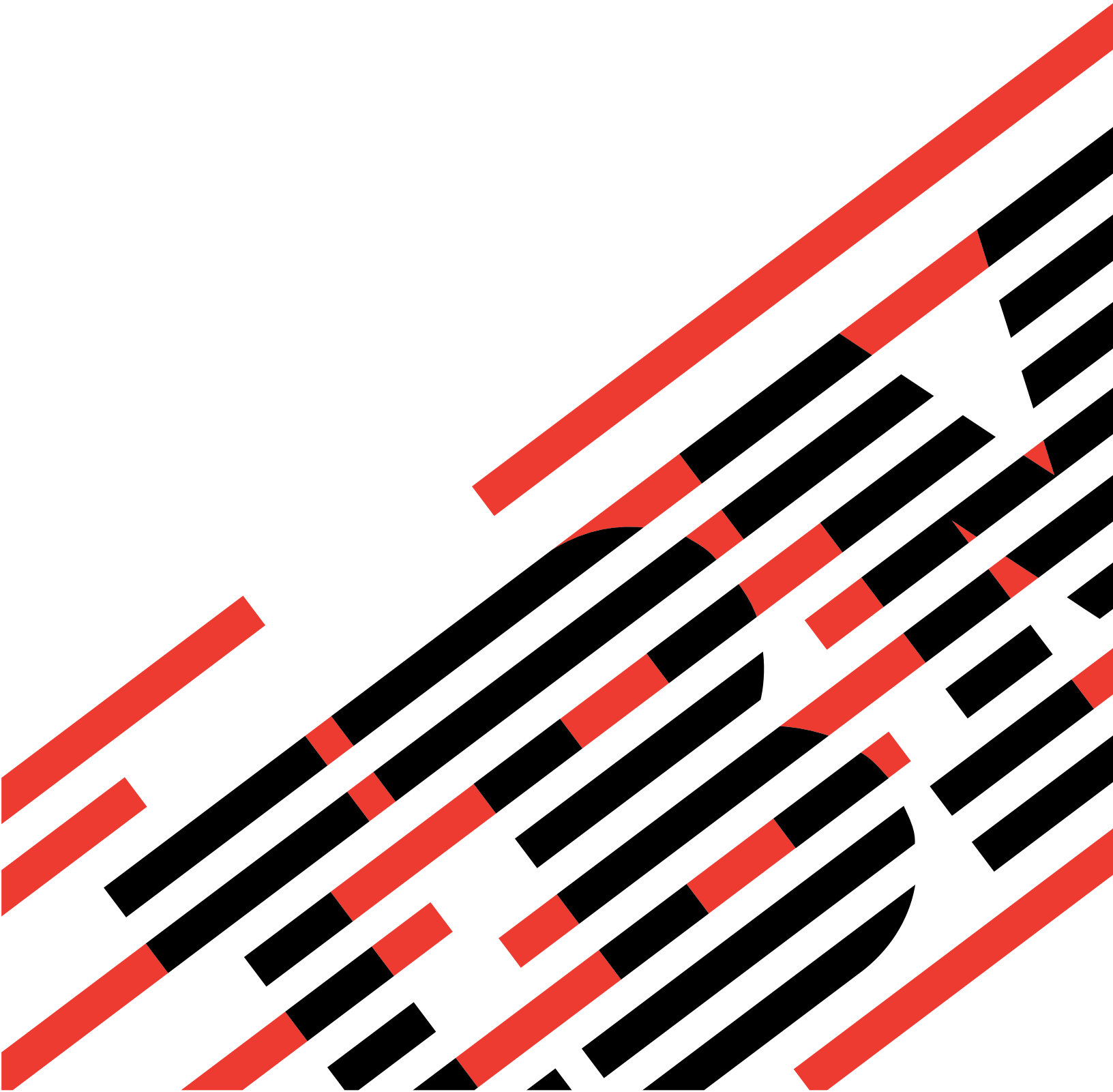
IBM Systems - iSeries

UNIX-Type -- Environment Variable APIs

Version 5 Release 4



IBM Systems - iSeries

UNIX-Type -- Environment Variable APIs

Version 5 Release 4

Note

Before using this information and the product it supports, be sure to read the information in "Notices," on page 37.

Sixth Edition (February 2006)

This edition applies to version 5, release 4, modification 0 of IBM i5/OS (product number 5722-SS1) and to all subsequent releases and modifications until otherwise indicated in new editions. This version does not run on all reduced instruction set computer (RISC) models nor does it run on CISC models.

Contents

Environment Variable APIs

Environment variables are character strings of the form ″name=value″. There are two types of environment variables:

- Job-level environment variables. The job-level environment variables are stored in an environment space outside of the program associated with the job. They can be manipulated by using the **getenv()**, **putenv()**, **Qp0zDltEnv()**, **Qp0zGetEnv()**, **Qp0zInitEnv()**, and **Qp0zPutEnv()** APIs, as well as the CL commands ADDENVVAR, CHGENVVAR, RMVENVVAR, and WRKENVVAR. These variables exist for the duration of the job or until they are deleted. There is a limit of 4095 job-level environment variables.
- System-level environment variables. The system-level environment variables are stored in a global environment space that is persistent across IPLs and is not associated to a particular job. They can be manipulated by using the **Qp0zDltSysEnv()**, **Qp0zGetAllSysEnv()**, **Qp0zGetSysEnv()**, and **Qp0zPutSysEnv()** APIs, as well as the CL commands ADDENVVAR, CHGENVVAR, RMVENVVAR, and WRKENVVAR. These variables exist until they are deleted. There is a limit of 4095 system-level environment variables.

When a job calls one of the job-level environment variable APIs or CL commands for the first time, it inherits the system-level environment variables onto its job-level environment space. Any changes to job-level and system-level environment variables are then independent of one another.

The temporary space where the job-level environment variables are stored allows read and write access. Therefore, it is possible for the space to be corrupted. This could occur if a programmer accesses the space directly using the `environ` array rather than using the environment variable APIs. If the space is corrupted, subsequent calls using the APIs will have unpredictable results.

The environment variable APIs are:

- “getenv()—Get Value of Environment Variable” on page 2 (Get value of environment variable) searches the job-level environment list for a string of the form name=value, where name is the environment variable and value is the value of the variable.
- “putenv()—Change or Add Environment Variable” on page 4 (Change or add environment variable) sets the value of a job-level environment variable by changing an existing variable or creating a new one.
- “Qp0zDltEnv()—Delete an Environment Variable” on page 7 (Delete an environment variable) deletes a single job-level environment variable or deletes all environment variables from the current job.
- “Qp0zDltSysEnv()—Delete a System-Level Environment Variable” on page 9 (Delete a system-level environment variable) deletes a single system-level environment variable or deletes all system-level environment variables.
- “Qp0zGetAllSysEnv()—Get All System-Level Environment Variables” on page 11 (Get all system-level environment variables) fills in the list_buf with a list of all the system-level environment variables.
- “Qp0zGetEnv()—Get Value of Environment Variable (Extended)” on page 13 (Get value of environment variable (extended)) is an i5/OS extension to the standard getenv() function.
- “Qp0zGetSysEnv()—Get Value of System-Level Environment Variable” on page 15 (Get value of system-level environment variable) gets the value of a system-level environment variable name by searching the system-level environment variable list for a string of the form name=value.
- “Qp0zInitEnv()—Initialize Environment for Variables” on page 17 (Initialize environment for variables) sets the external variable environ to a pointer to the current environment list.
- “Qp0zPutEnv()—Change or Add Environment Variable (Extended)” on page 18 (Change or add environment variable (extended)) is an i5/OS extension to the standard putenv() function.

- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21 (Change or add a system-level environment variable) sets the value of a system-level environment variable by altering an existing variable or creating a new variable.

Note: These functions use header (include) files from the library QSYSINC, which is optionally installable. Make sure QSYSINC is installed on your system before using any of the functions. See "Header Files for UNIX-Type Functions" on page 24 for the file and member name of each header file.

APIs

These are the APIs for this category.

getenv()—Get Value of Environment Variable

Syntax

```
#include <stdlib.h>

char *getenv(const char *name);
```

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes for more information.

The **getenv()** function searches the job-level environment list for a string of the form `name=value`, where `name` is the environment variable and `value` is the value of the variable.

The *name* parameter does not include the equal (=) symbol or the `value` of the environment variable `name=value` pair.

Authorities and Locks

None.

Parameters

name (Input) The name of an environment variable.

Return Value

value **getenv()** successfully found the environment string. The value returned is a pointer to the string containing the `value` for the specified `name` in the current environment.

NULL **getenv()** could not find the environment string. The *errno* variable is set to indicate the error.

Error Conditions

If **getenv()** is not successful, *errno* indicates one of the following errors.

[EDAMAGE]

A damaged object was encountered.

A referenced object is damaged. The object cannot be used.

[[EFAULT]]

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[ENOENT]

No such path or directory.

The directory or a component of the path name specified does not exist.

A named file or directory does not exist or is an empty string.

No entry found for name specified.

[EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

Usage Notes

1. Although **getenv()** is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the environ array the thread may see undefined results. The environ array can be accessed directly or by using a pointer returned from the **getenv()** or **Qp0zGetEnv()** functions. The environment contents are only protected during calls to the environment variable functions.
2. All environment variables are stored with an associated CCSID (coded character set identifier). Unless a different CCSID is specified, such as by using **Qp0zPutEnv()**, the default CCSID for the job is used as the CCSID associated with each environment variable string.
3. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information

- "putenv()—Change or Add Environment Variable" on page 4—Change or Add Environment Variable
- "Qp0zDltEnv()—Delete an Environment Variable" on page 7—Delete an Environment Variable
- "Qp0zDltSysEnv()—Delete a System-Level Environment Variable" on page 9—Delete a System-Level Environment Variable
- "Qp0zGetAllSysEnv()—Get All System-Level Environment Variables" on page 11—Get All System-Level Environment Variables
- "Qp0zGetEnv()—Get Value of Environment Variable (Extended)" on page 13—Get Value of Environment Variable (Extended)
- "Qp0zGetSysEnv()—Get Value of System-Level Environment Variable" on page 15—Get Value of System-Level Environment Variable
- "Qp0zInitEnv()—Initialize Environment for Variables" on page 17—Initialize Environment for Variables
- "Qp0zPutEnv()—Change or Add Environment Variable (Extended)" on page 18—Change or Add Environment Variable (Extended)
- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment

Example

See Code disclaimer information for information pertaining to code examples.

See the example of using **getenv()** in "putenv()—Change or Add Environment Variable"—Change or Add Environment Variable.

For other examples, see the following:
- Using Environment Variables
- Using the Spawn Process and Wait for Child Process APIs
- Using the Spawn Process (using NLS-enabled path name)

API introduced: V3R6

---

putenv()—Change or Add Environment Variable

Syntax

```
#include <stdlib.h>

int putenv(const char *string);;
```

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes for more information.

The **putenv()** function sets the value of a job-level environment variable by changing an existing variable or creating a new one. The *string* parameter points to a string of the form `name=value`, where `name` is the environment variable and `value` is the new value for it.

The `name` cannot contain a blank. For example,

```
PATH NAME=/my_lib/joe_user
```

is not valid because of the blank between `PATH` and `NAME`. The name can contain an equal (=) symbol, but the system interprets all characters following the first equal symbol as being the value of the environment variable. For example,

```
PATH=NAME=/my_lib/joe_user
```

will result in a value of 'NAME=/my_lib/joe_user' for the variable PATH.

Authorities and Locks

None.

Parameters

string (Input) A pointer to the `name=value` string.

Return Value

0 **putenv()** was successful.
-1 **putenv()** was not successful. The *errno* variable is set to indicate the error.

Error Conditions

If **putenv()** is not successful, *errno* indicates one of the following errors.

[EDAMAGE]

A damaged object was encountered.

A referenced object is damaged. The object cannot be used.

[EFAULT]

The address used for an argument is not correct. In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]

The value specified for the argument is not correct. A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL. For example, the string may not be in the correct format.

[ENOMEM]

Storage allocation request failed.

A function needed to allocate storage, but no storage is available.

There is not enough memory to perform the requested function. (There is a limit of 4095 environment variables per job.)

[EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

Usage Notes

1. Although **putenv()** is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the environ array the thread may see undefined results. The environ array can be accessed directly or by using a pointer returned from the **getenv()** or **Qp0zGetEnv()** functions. The environment contents are only protected during calls to the environment variable functions.
2. All environment variables are stored with an associated CCSID (coded character set identifier). Because **putenv()** does not specify a CCSID, the default CCSID for the job is used as the CCSID associated with strings that are stored using **putenv()**.
3. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information

- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "Qp0zDltEnv()—Delete an Environment Variable" on page 7—Delete an Environment Variable
- "Qp0zDltSysEnv()—Delete a System-Level Environment Variable" on page 9—Delete a System-Level Environment Variable
- "Qp0zGetAllSysEnv()—Get All System-Level Environment Variables" on page 11—Get All System-Level Environment Variables

- "Qp0zGetEnv()—Get Value of Environment Variable (Extended)" on page 13—Get Value of Environment Variable (Extended)
- "Qp0zGetSysEnv()—Get Value of System-Level Environment Variable" on page 15—Get Value of System-Level Environment Variable
- "Qp0zInitEnv()—Initialize Environment for Variables" on page 17—Initialize Environment for Variables
- "Qp0zPutEnv()—Change or Add Environment Variable (Extended)" on page 18—Change or Add Environment Variable (Extended)
- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment

Example

See Code disclaimer information for information pertaining to code examples.

The following example uses **putenv()** and **getenv()**.

```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

int main(int argc, char **argv)
{
 char     *var1  = "PATH=/:/home/userid";
 char     *name1 = "PATH";
 char     *val1  = NULL;
 int      rc;

 rc = putenv(var1);
 if (rc < 0) {
   printf("Error inserting <%s> in environ, errno = %d\n",
           var1, errno);
   return 1;
 }

 printf("<%s> inserted in environ\n", var1);
 val1 = getenv(name1);
 if (val1 == NULL) {
   printf("Error retrieving <%s> from environ, errno = %d\n",
           name1, errno);
   return 1;
 }

 printf("<%s> retrieved from environ, value is <%s>\n",
         name1, val1);
 return 0;
}
```

Output:

```
 <PATH=/:/home/userid> inserted in environ
 <PATH> retrieved from environ, value is </:/home/userid>
```

For other examples, see the following:

- Using Environment Variables.
- Using the Spawn Process and Wait for Child Process APIs.
- Using the Spawn Process (using NLS-enabled path name)

API introduced: V3R6

Top | UNIX-Type APIs | APIs by category

Qp0zDltEnv()—Delete an Environment Variable

Syntax

```
#include <qp0z1170.h>

int Qp0zDltEnv(const char *name);
```

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes for more information.

The **Qp0zDltEnv()** function deletes a single job-level environment variable or deletes all environment variables from the current job. If the *name* parameter is NULL, all environment variables in the job are deleted.

The *name* parameter does not include the equal (=) symbol or the `value` of the environment variable `name=value` pair.

Parameters

name (Input) A pointer to the `name` part of the environment variable `name=value` string.

Authorities

None.

Return Value

0 **Qp0zDltEnv()** was successful.
-1 **Qp0zDltEnv()** was not successful. The *errno* variable is set to indicate the error.

Error Conditions

If **Qp0zDltEnv()** is not successful, *errno* indicates one of the following errors.

[ENOENT]

No such path or directory.

The directory or a component of the path name specified does not exist.

A named file or directory does not exist or is an empty string.

The parameter `name` is not NULL and does not point to an environment variable name that currently exists in the environment list.

Usage Notes

1. Although **Qp0zDltEnv()** is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the environ array the thread may see undefined results. The environ array can be accessed directly or by using a pointer returned from the **getenv()** or **Qp0zGetEnv()** functions. The environment contents are only protected during calls to the environment variable functions.

Related Information

- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "putenv()—Change or Add Environment Variable" on page 4—Change or Add Environment Variable

- “Qp0zDltSysEnv()—Delete a System-Level Environment Variable” on page 9—Delete a System-Level Environment Variable
- “Qp0zGetAllSysEnv()—Get All System-Level Environment Variables” on page 11)—Get All System-Level Environment Variables
- “Qp0zGetEnv()—Get Value of Environment Variable (Extended)” on page 13—Get Value of Environment Variable (Extended)
- “Qp0zGetSysEnv()—Get Value of System-Level Environment Variable” on page 15—Get Value of System-Level Environment Variable
- “Qp0zInitEnv()—Initialize Environment for Variables” on page 17—Initialize Environment for Variables
- “Qp0zPutEnv()—Change or Add Environment Variable (Extended)” on page 18—Change or Add Environment Variable (Extended)
- “Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable” on page 21—Change or Add a System-Level Environment

Example

See Code disclaimer information for information pertaining to code examples.

The following example uses **Qp0zDltEnv()**, **putenv()** and the environ array.

```
#include <stdio.h>
#include <errno.h>
#include <qp0z1170.h>
#include <stdlib.h>

extern char **environ;

#define ASSERT(x, y)                        \
{ if (!(x)) {                               \
    printf("Assertion Failed: " #x          \
           ", Description: " y              \
           ", errno=%d", errno);            \
    exit(EXIT_FAILURE);                     \
  }                                         \
}

int main(int argc, char **argv)
{
  int                   rc=0;
  int                   e=0;
  printf("Enter Testcase - %s\n", argv[0]);

  rc = putenv("PATH=/usr/bin:/home/me:%LIBL%");
  ASSERT((rc == 0), "putenv(PATH)");
  rc = putenv("TEST0=42");
  ASSERT((rc == 0), "putenv(TEST0)");
  rc = putenv("TEST1=42");
  ASSERT((rc == 0), "putenv(TEST1)");
  printf("Before delete, these environment variables are set: \n");

  while (environ[e] != NULL) {
    printf("  %s\n", environ[e]);
    ++e;
  }

  printf("Delete the environment variables\n");
  rc = Qp0zDltEnv("TEST0");
  ASSERT((rc==0), "Qp0zDltEnv(TEST0)");
  rc = Qp0zDltEnv("TEST1");
  ASSERT((rc==0), "Qp0zDltEnv(TEST1)");

  printf("After delete, these environment variables are set: \n");
```

```
  e=0;
  while (environ[e] != NULL) {
    printf("  %s\n", environ[e]);
    ++e;
  }
  printf("Main completed\n");
  return 0;
}
```

Output:

```
Enter Testcase - QP0WTEST/TPZDLTE0
Before delete, these environment variables are set:
  PATH=/usr/bin:/home/me:%LIBL%
  TEST0=42
  TEST1=42
Delete the environment variables
After delete, these environment variables are set:
  PATH=/usr/bin:/home/me:%LIBL%
Main completed
```

API introduced: V4R3

Top | UNIX-Type APIs | APIs by category

Qp0zDltSysEnv()—Delete a System-Level Environment Variable

Syntax

```
#include <qp0z1170.h>

int Qp0zDltSysEnv(const char *name, void  *reserved);
```

Service Program Name: QP0ZSYSE
Default Public Authority: *USE
Threadsafe: Yes

The **Qp0zDltSysEnv()** function deletes a single system-level environment variable or deletes all system-level environment variables. If the *name* parameter is NULL, all system-level environment variables are deleted.

The *name* parameter does not include the equal (=) symbol or the `value` part of the environment variable `name=value` pair.

Parameters

name (Input) The name of the environment variable to delete.

reserved
(Input) Reserved for future use. Currently, the only value allowed is NULL.

Authorities

*JOBCTL special authority is required to delete a system-level environment variable.

Return Value

0 **Qp0zDltSysEnv()** was successful.
errval **Qp0zDltSysEnv()** was not successful. *errval* is set to indicate the error.

Error Conditions

If **Qp0zDltSysEnv()** is not successful, *errval* indicates one of the following errors.

[EFAULT]

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL.

The value for the *reserved* parameter was not NULL.

[ENOENT]

No such path or directory.

The directory or a component of the path name specified does not exist.

A named file or directory does not exist or is an empty string.

The parameter `name` is not NULL and does not point to an environment variable name that currently exists in the environment list.

[EPERM]

Operation not permitted.

You must have appropriate privileges or be the owner of the object or other resource to do the requested operation.

You must have *JOBCTL special authority to delete a system-level environment variable.

[EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

Related Information

- The <**qp0z1170.h**> file (see "Header Files for UNIX-Type Functions" on page 24)
- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "putenv()—Change or Add Environment Variable" on page 4—Change or Add Environment Variable
- "Qp0zDltEnv()—Delete an Environment Variable" on page 7—Delete an Environment Variable
- "Qp0zGetAllSysEnv()—Get All System-Level Environment Variables" on page 11—Get All System-Level Environment Variables
- "Qp0zGetEnv()—Get Value of Environment Variable (Extended)" on page 13—Get Value of Environment Variable (Extended)
- "Qp0zGetSysEnv()—Get Value of System-Level Environment Variable" on page 15—Get Value of System-Level Environment Variable
- "Qp0zInitEnv()—Initialize Environment for Variables" on page 17—Initialize Environment for Variables

- "Qp0zPutEnv()—Change or Add Environment Variable (Extended)" on page 18—Change or Add Environment Variable (Extended)
- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment

Example

See Code disclaimer information for information pertaining to code examples.

See the example of using **Qp0zDltSysEnv()** in "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment.

API introduced: V4R4

Top | UNIX-Type APIs | APIs by category

Qp0zGetAllSysEnv()—Get All System-Level Environment Variables

Syntax

```
#include <qp0z1170.h>

int Qp0zGetAllSysEnv(char *list_buf, int  *list_buf_size,
                 int *ccsid_buf, int  *ccsid_buf_size,
                 void *reserved);
```

Service Program Name: QP0ZSYSE
Default Public Authority: *USE
Threadsafe: Yes

The **Qp0zGetAllSysEnv()** function fills in the `list_buf` with a list of all the system-level environment variables. The list consists of multiple null-terminated `name=value` strings followed by an ending null-terminator. The coded character set identifier (CCSID) associated with each `name=value` string is returned in the `ccsid_buf` buffer.

Authorities

None

Parameters

list_buf
(Input/Output) The address of the buffer to receive the null-terminated `name=value` list.

list_buf_size
(Input/Output) A pointer to an integer that contains the information about the size (in bytes) of the `list_buf` buffer. Before calling **Qp0zGetAllSysEnv()**, this parameter should be set to the size of `list_buf`. If the size of this parameter is large enough to receive the list, then this field will be set to the exact size of the list upon returning from **Qp0zGetAllSysEnv().** If the size of this parameter is not large enough to receive the list, then this field will contain the exact size required and ENOSPC will be the return value. In this case, the *list_buf* is not modified.

ccsid_buf
(Input/Output) The address of the buffer to receive the CCSIDs of the environment variables. The order of the CCSIDs returned corresponds to the order of the variables returned in the *list_buf*

ccsid_buf_size
(Input/Output) A pointer to an integer that contains the information about the size (in bytes) of the `ccsid_buf` buffer. Before calling **Qp0zGetAllSysEnv()**, this should be set to the size of

`ccsid_buf`. If this size is enough to receive the CCSID list, then this field will contain the exact size of the CCSIDs received upon returning from **Qp0zGetAllSysEnv()**. If this size is not enough to receive the CCSID list, then this field will contain the exact size required and ENOSPC will be the return value. In this case, the *ccsid_buf* is not modified.

reserved
(Input) Reserved for future use. Currently, the only allowed value is NULL.

Return Value

0	**Qp0zGetAllSysEnv()** was successful. The `list_buf` contains the null-terminated system-level environment variable strings, and the `ccsid_buf` contains the CCSID of each variable in the same order. The `list_buf_size` contains the exact size of the environment variable list, and the `ccsid_buf_size` contains the exact size of the CCSID list.
errval	**Qp0zGetAllSysEnv()** was not succesful. *errval* indicates the error.

Error Conditions

If **Qp0zGetAllSysEnv()** is not successful, *errval* indicates one of the following errors.

[EFAULT]

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL.

The value for the *reserved* parameter was not NULL.

[ENOENT]

No such path or directory.

The directory or a component of the path name specified does not exist.

A named file or directory does not exist or is an empty string.

There were no system-level environment variables.

[ENOSPC]

No space available.

The requested operations required additional space on the device and there is no space left. This could also be caused by exceeding the user profile storage limit when creating or transferring ownership of an object.

Insufficient space remains to hold the intended file, directory, or link.

The size of the buffers to receive the list and the CCSIDs was not enough. The *list_buf_size* and *ccsid_buf_size* parameters indicate the exact size needed for the *list_buf ccsid_buf* respectively.

[EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

Usage Notes

1. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information

- The <**qp0z1170.h**> file (see "Header Files for UNIX-Type Functions" on page 24)
- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "putenv()—Change or Add Environment Variable" on page 4—Change or Add Environment Variable
- "Qp0zDltEnv()—Delete an Environment Variable" on page 7—Delete an Environment Variable
- "Qp0zDltSysEnv()—Delete a System-Level Environment Variable" on page 9—Delete a System-Level Environment Variable
- "Qp0zGetEnv()—Get Value of Environment Variable (Extended)"—Get Value of Environment Variable (Extended)
- "Qp0zGetSysEnv()—Get Value of System-Level Environment Variable" on page 15—Get Value of System-Level Environment Variable
- "Qp0zInitEnv()—Initialize Environment for Variables" on page 17—Initialize Environment for Variables
- "Qp0zPutEnv()—Change or Add Environment Variable (Extended)" on page 18—Change or Add Environment Variable (Extended)
- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment

Example

See Code disclaimer information for information pertaining to code examples.

1. See the example in "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment.
2. See the two-part example in Saving and Restoring System-Level Environment Variables in Examples: APIS.

API introduced: V4R4

Top | UNIX-Type APIs | APIs by category

Qp0zGetEnv()—Get Value of Environment Variable (Extended)

Syntax

```
#include <qp0z1170.h>

char *Qp0zGetEnv(const char *name, int *ccsid);
```

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes for more information.

The **Qp0zGetEnv()** function is an i5/OS extension to the standard **getenv()** function. **Qp0zGetEnv()** searches the job-level environment list for a string of the form `name=value`. The `value` and the CCSID (coded character set identifier) associated with the environment variable `name` are returned.

Authorities and Locks

None.

Parameters

name (Input) The name of an environment variable.

ccsid (Output) The CCSID for the named environment variable.

Return Value

value	**Qp0zGetEnv()** successfully found the environment string. The value returned is a pointer to the string containing the `value` for the specified `name` in the current environment.
NULL	**Qp0zGetEnv()** could not find the environment string. The *errno* variable is set to indicate the error.

Error Conditions

If **Qp0zGetEnv()** is not successful, *errno* indicates one of the following errors.

[EDAMAGE]

A damaged object was encountered.

A referenced object is damaged. The object cannot be used.

[EFAULT]

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[ENOENT]

No such path or directory.

The directory or a component of the path name specified does not exist.

A named file or directory does not exist or is an empty string.

No entry found for name specified.

[EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

Usage Notes

1. Although **Qp0zGetEnv()** is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the environ array the thread may see undefined results. The environ array can be accessed directly or by using a pointer returned from the **getenv()** or **Qp0zGetEnv()** functions. The environment contents are only protected during calls to the environment variable functions.
2. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information

- The <**qp0z1170.h**> file (see "Header Files for UNIX-Type Functions" on page 24)
- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "putenv()—Change or Add Environment Variable" on page 4—Change or Add Environment Variable
- "Qp0zDltEnv()—Delete an Environment Variable" on page 7—Delete an Environment Variable
- "Qp0zDltSysEnv()—Delete a System-Level Environment Variable" on page 9—Delete a System-Level Environment Variable
- "Qp0zGetAllSysEnv()—Get All System-Level Environment Variables" on page 11—Get All System-Level Environment Variables
- "Qp0zGetSysEnv()—Get Value of System-Level Environment Variable"—Get Value of System-Level Environment Variable
- "Qp0zInitEnv()—Initialize Environment for Variables" on page 17—Initialize Environment for Variables
- "Qp0zPutEnv()—Change or Add Environment Variable (Extended)" on page 18—Change or Add Environment Variable (Extended)
- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment

Example

See Code disclaimer information for information pertaining to code examples.

See the example of using **getenv()** in "putenv()—Change or Add Environment Variable" on page 4—Change or Add Environment Variable.

API introduced: V3R6

Top | UNIX-Type APIs | APIs by category

Qp0zGetSysEnv()—Get Value of System-Level Environment Variable

Syntax

```
#include <qp0z1170.h>

int Qp0zGetSysEnv(const char *name,
                  char *value, int *value_size,
                  int *ccsid, void *reserved);
```

Service Program Name: QP0ZSYSE
Default Public Authority: *USE
Threadsafe: Yes

The **Qp0zGetSysEnv()** function gets the `value` of a system-level environment variable `name` by searching the system-level environment variable list for a string of the form `name=value`. The `value` and the coded character set identifier (CCSID) associated with the environment variable `name` are returned.

Authorities

None

Parameters

name (Input) The name of an environment variable.

value (Input/Output) The address of the buffer to receive the `value`.

value_size
(Input/Output) A pointer to an integer that contains the information about the size of the `value` buffer. Before calling **Qp0zGetSysEnv()**, this parameter should contain the size of the `value` buffer. If the size of this parameter is large enough to receive the `value`, then this field will contain the exact size of `value` upon returning from **Qp0zGetSysEnv()**. If the size of this parameter is not large enough to receive the `value`, then this field will contain the exact size required and ENOSPC will be the return value. In this case, the `value` buffer is not modified.

ccsid (Input/Output) The address of the variable to receive the CCSID associated with this variable.

reserved
(Input) Reserved for future use. Currently, the only allowed value is NULL.

Return Value

0	**Qp0zGetSysEnv()** successfully found the environment string. `value` and `ccsid` contain the value and CCSID for the variable `name` in the system-level environment variable list.
errval	**Qp0zGetEnv()** was not successful. *errval* indicates the error./td>

Error Conditions

If **Qp0zGetSysEnv()** is not successful, *errval* indicates one of the following errors.

[EFAULT]

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL.

The value for the *reserved* parameter was not NULL.

[ENOENT]

No such path or directory.

The directory or a component of the path name specified does not exist.

A named file or directory does not exist or is an empty string.

No entry found for name specified.

[ENOSPC]

No space available.

The requested operations required additional space on the device and there is no space left. This could also be caused by exceeding the user profile storage limit when creating or transferring ownership of an object.

Insufficient space remains to hold the intended file, directory, or link.

The size of the *value* buffer was not big enough to receive the value.

[EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

Usage Notes

1. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information

- The <**qp0z1170.h**> file (see "Header Files for UNIX-Type Functions" on page 24)
- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "putenv()—Change or Add Environment Variable" on page 4—Change or Add Environment Variable
- "Qp0zDltEnv()—Delete an Environment Variable" on page 7—Delete an Environment Variable
- "Qp0zDltSysEnv()—Delete a System-Level Environment Variable" on page 9—Delete a System-Level Environment Variable
- "Qp0zGetAllSysEnv()—Get All System-Level Environment Variables" on page 11—Get All System-Level Environment Variables
- "Qp0zGetEnv()—Get Value of Environment Variable (Extended)" on page 13—Get Value of Environment Variable (Extended)
- "Qp0zInitEnv()—Initialize Environment for Variables"—Initialize Environment for Variables
- "Qp0zPutEnv()—Change or Add Environment Variable (Extended)" on page 18—Change or Add Environment Variable (Extended)
- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment

Example

See Code disclaimer information for information pertaining to code examples.

See the example of using **Qp0zGetSysEnv()** in "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment.

API introduced: V4R4

Top | UNIX-Type APIs | APIs by category

Qp0zInitEnv()—Initialize Environment for Variables

Syntax

```
#include <qp0z1170.h>

int Qp0zInitEnv(void);;
```

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes

The **Qp0zInitEnv()** function sets the external variable `environ` to a pointer to the current environment list. (On the iSeries server, `environ` is initialized to NULL when an activation group is started.)

Note: Although it is possible for a user's program to directly read the `environ` array, use of the **getenv()** or **Qp0zGetEnv()** functions is recommended.

Authorities and Locks

None.

Parameters

None.

Return Value

0 **Qp0zInitEnv()** successfully initialized the environment.
-1 **Qp0zInitEnv()** was not successful. The *errno* variable is set to indicate the error.

Error Conditions

If **Qp0zInitEnv()** is not successful, *errno* indicates the following error.

[EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

Related Information

- The <**qp0z1170.h**> file (see "Header Files for UNIX-Type Functions" on page 24)
- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "putenv()—Change or Add Environment Variable" on page 4—Change or Add Environment Variable
- "Qp0zDltEnv()—Delete an Environment Variable" on page 7—Delete an Environment Variable
- "Qp0zDltSysEnv()—Delete a System-Level Environment Variable" on page 9—Delete a System-Level Environment Variable
- "Qp0zGetAllSysEnv()—Get All System-Level Environment Variables" on page 11—Get All System-Level Environment Variables
- "Qp0zGetEnv()—Get Value of Environment Variable (Extended)" on page 13—Get Value of Environment Variable (Extended)
- "Qp0zGetSysEnv()—Get Value of System-Level Environment Variable" on page 15—Get Value of System-Level Environment Variable
- "Qp0zPutEnv()—Change or Add Environment Variable (Extended)"—Change or Add Environment Variable (Extended)
- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment

API introduced: V3R6

Top | UNIX-Type APIs | APIs by category

Qp0zPutEnv()—Change or Add Environment Variable (Extended)

Syntax

```
#include <qp0z1170.h>

int Qp0zPutEnv(const char *string, int ccsid);;
```

Service Program Name: QP0ZCPA
Default Public Authority: *USE
Threadsafe: Yes. See Usage Notes for more information.

The **Qp0zPutEnv()** function is an i5/OS extension to the standard **putenv()** function. **Qp0zPutEnv()** sets the value of an environment variable by altering an existing variable or creating a new variable. In addition, it specifies a CCSID (coded character set identifier) to be associated with the environment variable.

The *string* parameter points to a string of the form `name=value`, where `name` is the environment variable and `value` is the new value for it.

The `name` cannot contain a blank. For example,

```
PATH NAME=/my_lib/joe_user
```

is not valid because of the blank between `PATH` and `NAME`. The name can contain an equal (=) symbol, but the system interprets all characters following the first equal symbol as being the value of the environment variable. For example,

```
PATH=NAME=/my_lib/joe_user
```

will result in a value of 'NAME=/my_lib/joe_user' for the variable PATH.

Authorities and Locks

None.

Parameters

string (Input) A pointer to the `name=value` string.

ccsid (Input) A CCSID to be associated with this environment variable. If 0 is specified, the default CCSID for the job is used.

Return Value

0 **Qp0zInitEnv()** successfully initialized the environment.
-1 **Qp0zInitEnv()** was not successful. The *errno* variable is set to indicate the error.

Error Conditions

If **Qp0zPutEnv()** is not successful, *errno* indicates one of the following errors.

[EDAMAGE]

A damaged object was encountered.

A referenced object is damaged. The object cannot be used.

[EFAULT]

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL. For example, the string may not be in the correct format.

[ENOMEM]

Storage allocation request failed.

A function needed to allocate storage, but no storage is available.

There is not enough memory to perform the requested function. (There is a limit of 4095 environment variables per job.)

[EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

Usage Notes

1. Although **Qp0zPutEnv()** is threadsafe, if a thread calls an environment variable function while another thread is accessing an environment variable from the environ array the thread may see undefined results. The environ array can be accessed directly or by using a pointer returned from the **getenv()** or **Qp0zGetEnv()** functions. The environment contents are only protected during calls to the environment variable functions.
2. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information

- The <**qp0z1170.h**> file (see "Header Files for UNIX-Type Functions" on page 24)
- "getenv()—Get Value of Environment Variable" on page 2—Get Value of Environment Variable
- "putenv()—Change or Add Environment Variable" on page 4—Change or Add Environment Variable
- "Qp0zDltEnv()—Delete an Environment Variable" on page 7—Delete an Environment Variable
- "Qp0zDltSysEnv()—Delete a System-Level Environment Variable" on page 9—Delete a System-Level Environment Variable
- "Qp0zGetAllSysEnv()—Get All System-Level Environment Variables" on page 11—Get All System-Level Environment Variables
- "Qp0zGetSysEnv()—Get Value of System-Level Environment Variable" on page 15—Get Value of System-Level Environment Variable
- "Qp0zInitEnv()—Initialize Environment for Variables" on page 17—Initialize Environment for Variables
- "Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable" on page 21—Change or Add a System-Level Environment Variable

Example

See Code disclaimer information for information pertaining to code examples.

See the example of using **putenv()** in "Qp0zPutEnv()—Change or Add Environment Variable (Extended)" on page 18—Change or Add Environment Variable.

API introduced: V3R6

Qp0zPutSysEnv()—Change or Add a System-Level Environment Variable

Syntax

```
#include <qp0z1170.h>

int Qp0zPutSysEnv(const char *string, int ccsid,
                  void *reserved);
```

Service Program Name: QP0ZSYSE
Default Public Authority: *USE
Threadsafe: Yes

Qp0zPutSysEnv() function sets the value of a system-level environment variable by altering an existing variable or creating a new variable. In addition, it specifies a CCSID (coded character set identifier) to be associated with the environment variable.

The *string* parameter points to a string of the form `name=value`, where `name` is the environment variable and `value` is the new value for it.

The `name` cannot contain a blank. For example,

```
PATH NAME=/my_lib/joe_user
```

is not valid because of the blank between `PATH` and `NAME`. The name can contain an equal (=) symbol, but the system interprets all characters following the first equal symbol as being the value of the environment variable. For example,

```
PATH=NAME=/my_lib/joe_user
```

will result in a value of 'NAME=/my_lib/joe_user' for the variable PATH.

Parameters

string (Input) A pointer to the `name=value` string.

ccsid (Input) A CCSID to be associated with this environment variable. If 0 is specified, the default CCSID for the job is used.

reserved
(Input) Reserved for future use. Currently, the only allowed value is NULL.

Authorities

*JOBCTL special authority is required to add or change a system-level environment variable.

Return Value

0	**Qp0zPutSysEnv()** was successful.
errval	**Qp0zPutSysEnv()** was not successful. *errval* is set to indicate the error.

Error Conditions

If **Qp0zPutSysEnv()** is not successful, *errval* indicates one of the following errors.

[EFAULT]

The address used for an argument is not correct.

In attempting to use an argument in a call, the system detected an address that is not valid.

While attempting to access a parameter passed to this function, the system detected an address that is not valid.

[EINVAL]

The value specified for the argument is not correct.

A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.

An argument value is not valid, out of range, or NULL.

For example, the *string* parameter was not in the correct format or the value for the *reserved* parameter was not NULL.

[ENOMEM]

Storage allocation request failed.

A function needed to allocate storage, but no storage is available.

There is not enough memory to perform the requested function. (There is a limit of 4095 system-level environment variables.)

[EOPNOTSUPP]

Operation not supported.

The operation, though supported in general, is not supported for the requested object or the requested arguments.

This error is returned if the environment variable that is being added is QIBM_CHILD_JOB_SNDINQMSG. See **spawn()** in or **spawnp()** in for details on QIBM_CHILD_JOB_SNDINQMSG.

[EPERM]

Operation not permitted.

You must have appropriate privileges or be the owner of the object or other resource to do the requested operation.

You must have *JOBCTL special authority to add or change system-level environment variables.

[EUNKNOWN]

Unknown system state.

The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.

Usage Notes

1. No translation is done based on the CCSID. The CCSID is just stored and retrieved as an integer value associated with each environment variable.

Related Information

- The <**qp0z1170.h**> file (see “Header Files for UNIX-Type Functions” on page 24)
- “getenv()—Get Value of Environment Variable” on page 2—Get Value of Environment Variable
- “putenv()—Change or Add Environment Variable” on page 4—Change or Add Environment Variable
- “Qp0zDltEnv()—Delete an Environment Variable” on page 7—Delete an Environment Variable
- “Qp0zDltSysEnv()—Delete a System-Level Environment Variable” on page 9—Delete a System-Level Environment Variable

- "Qp0zGetAllSysEnv()—Get All System-Level Environment Variables" on page 11—Get All System-Level Environment Variables
- "Qp0zGetEnv()—Get Value of Environment Variable (Extended)" on page 13—Get Value of Environment Variable (Extended)
- "Qp0zGetSysEnv()—Get Value of System-Level Environment Variable" on page 15—Get Value of System-Level Environment Variable
- "Qp0zInitEnv()—Initialize Environment for Variables" on page 17—Initialize Environment for Variables
- "Qp0zPutEnv()—Change or Add Environment Variable (Extended)" on page 18—Change or Add Environment Variable (Extended)

Example

See Code disclaimer information for information pertaining to code examples.

The following example uses **Qp0zPutSysEnv()**, **Qp0zGetSysEnv()**, and **Qp0zDltSysEnv()**.

```
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <qp0z1170.h>

int main(int argc, char **argv)
{
 char     *var1  = "PATH=/:/home";
 char     *name1 = "PATH";
 char     *val1  = NULL;
 int      rc, ccsid, size;

 /* Add the system-level variable PATH */
 /* using default ccsid               */
 ccsid = 0;
 rc = Qp0zPutSysEnv(var1, ccsid, NULL);
 if(rc != 0)
 {
   printf("Error from Qp0zPutSysEnv while adding <%s>\n",var1);
   printf("errno = %d\n",rc);
   return rc;
 }

 printf("<%s> added to system-level env var list\n",var1);

 /* Get the value of the variable PATH */
 size = 100;
 val1 = (char *)malloc(size);

 rc = Qp0zGetSysEnv(name1, val1, &size, &ccsid, NULL);
 if(rc == ENOSPC)
 {
  /* The buffer size was not enough to get the value */
  /* Increase the buffer to size                     */
  val1 = (char *)realloc(val1, size);
  rc = Qp0zGetSysEnv(name1, val1, &size, &ccsid, NULL);
 }

 if(rc != 0)
 {
  printf("Error from Qp0zGetSysEnv while retrieving");
  printf("<%s>, errno = %d\n", name1, rc);
  return rc;
 }

 printf("<%s> retrieved, value is <%s>\n",name1,val1);

 /* Delete the PATH variable */
```

```
 rc = Qp0zDltSysEnv(name1, NULL);
 if(rc != 0)
 {
  printf("Error from Qp0zDltSysEnv while deleting");
  printf("<%s>, errno = %d\n", name1, rc);
  return rc;
 }

 printf("<%s> deleted from system-level env var list\n",name1);

 return 0;
}
```

Output:

```
<PATH=/:/home> added to system-level variable list
<PATH> retrieved, value is </:/home>
<PATH> deleted from system-level variable list
```

For other examples, see the two-part example in API Examples for saving and restoring system-level environment variables.

API introduced: V4R4

Top | UNIX-Type APIs | APIs by category

Concepts

These are the concepts for this category.

Header Files for UNIX-Type Functions

Programs using the UNIX(R)-type functions must include one or more header files that contain information needed by the functions, such as:

- Macro definitions
- Data type definitions
- Structure definitions
- Function prototypes

The header files are provided in the QSYSINC library, which is optionally installable. Make sure QSYSINC is on your system before compiling programs that use these header files. For information on installing the QSYSINC library, see Include files and the QSYSINC Library.

The table below shows the file and member name in the QSYSINC library for each header file used by the UNIX-type APIs in this publication.

Name of Header File	Name of File in QSYSINC	Name of Member
arpa/inet.h	ARPA	INET
arpa/nameser.h	ARPA	NAMESER
bse.h	H	BSE
bsedos.h	H	BSEDOS
bseerr.h	H	BSEERR
dirent.h	H	DIRENT
errno.h	H	ERRNO

Name of Header File	Name of File in QSYSINC	Name of Member
fcntl.h	H	FCNTL
grp.h	H	GRP
inttypes.h	H	INTTYPES
limits.h	H	LIMITS
mman.h	H	MMAN
netdbh.h	H	NETDB
netinet/icmp6.h	NETINET	ICMP6
net/if.h	NET	IF
netinet/in.h	NETINET	IN
netinet/ip_icmp.h	NETINET	IP_ICMP
netinet/ip.h	NETINET	IP
netinet/ip6.h	NETINET	IP6
netinet/tcp.h	NETINET	TCP
netinet/udp.h	NETINET	UDP
netns/idp.h	NETNS	IDP
netns/ipx.h	NETNS	IPX
netns/ns.h	NETNS	NS
netns/sp.h	NETNS	SP
net/route.h	NET	ROUTE
nettel/tel.h	NETTEL	TEL
os2.h	H	OS2
os2def.h	H	OS2DEF
pwd.h	H	PWD
Qlg.h	H	QLG
qp0lchsg.h	H	QP0LCHSG
qp0lflop.h	H	QP0LFLOP
qp0ljrnl.h	H	QP0LJRNL
qp0lror.h	H	QP0LROR
qp0lrro.h	H	QP0LRRO
qp0lrtsg.h	H	QP0LRTSG
qp0lscan.h	H	QP0LSCAN
Qp0lstdi.h	H	QP0LSTDI
qp0wpid.h	H	QP0WPID
qp0zdipc.h	H	QP0ZDIPC
qp0zipc.h	H	QP0ZIPC
qp0zolip.h	H	QP0ZOLIP
qp0zolsm.h	H	QP0ZOLSM
qp0zripc.h	H	QP0ZRIPC
qp0ztrc.h	H	QP0ZTRC
qp0ztrml.h	H	QP0ZTRML
qp0z1170.h	H	QP0Z1170

Name of Header File	Name of File in QSYSINC	Name of Member
qsoasync.h	H	QSOASYNC
qtnxaapi.h	H	QTNXAAPI
qtnxadtp.h	H	QTNXADTP
qtomeapi.h	H	QTOMEAPI
qtossapi.h	H	QTOSSAPI
resolv.h	H	RESOLVE
semaphore.h	H	SEMAPHORE
signal.h	H	SIGNAL
spawn.h	H	SPAWN
ssl.h	H	SSL
sys/errno.h	H	ERRNO
sys/ioctl.h	SYS	IOCTL
sys/ipc.h	SYS	IPC
sys/layout.h	H	LAYOUT
sys/limits.h	H	LIMITS
sys/msg.h	SYS	MSG
sys/param.h	SYS	PARAM
sys/resource.h	SYS	RESOURCE
sys/sem.h	SYS	SEM
sys/setjmp.h	SYS	SETJMP
sys/shm.h	SYS	SHM
sys/signal.h	SYS	SIGNAL
sys/socket.h	SYS	SOCKET
sys/stat.h	SYS	STAT
sys/statvfs.h	SYS	STATVFS
sys/time.h	SYS	TIME
sys/types.h	SYS	TYPES
sys/uio.h	SYS	UIO
sys/un.h	SYS	UN
sys/wait.h	SYS	WAIT
ulimit.h	H	ULIMIT
unistd.h	H	UNISTD
utime.h	H	UTIME

You can display a header file in QSYSINC by using one of the following methods:

- Using your editor. For example, to display the **unistd.h** header file using the Source Entry Utility editor, enter the following command:

  ```
  STRSEU SRCFILE(QSYSINC/H) SRCMBR(UNISTD) OPTION(5)
  ```

- Using the Display Physical File Member command. For example, to display the **sys/stat.h** header file, enter the following command:

```
DSPPFM FILE(QSYSINC/SYS) MBR(STAT)
```

You can print a header file in QSYSINC by using one of the following methods:

- Using your editor. For example, to print the **unistd.h** header file using the Source Entry Utility editor, enter the following command:

  ```
  STRSEU SRCFILE(QSYSINC/H) SRCMBR(UNISTD) OPTION(6)
  ```

- Using the Copy File command. For example, to print the **sys/stat.h** header file, enter the following command:

  ```
  CPYF FROMFILE(QSYSINC/SYS) TOFILE(*PRINT) FROMMBR(STAT)
  ```

Symbolic links to these header files are also provided in directory /QIBM/include.

Top | UNIX-Type APIs | APIs by category

Errno Values for UNIX-Type Functions

Programs using the UNIX(R)-type functions may receive error information as *errno* values. The possible values returned are listed here in ascending *errno* value sequence.

Name	Value	Text	Details
EDOM	3001	A domain error occurred in a math function.	
ERANGE	3002	A range error occurred.	
ETRUNC	3003	Data was truncated on an input, output, or update operation.	
ENOTOPEN	3004	File is not open.	You attempted to do an operation that required the file to be open.
ENOTREAD	3005	File is not opened for read operations.	You tried to read a file that is not open for read operations.
EIO	3006	Input/output error.	» A physical I/O error occurred or a referenced object was damaged. «
ENODEV	3007	No such device.	
ERECIO	3008	Cannot get single character for files opened for record I/O.	The file that was specified is open for record I/O and you attempted to read it as a stream file.
ENOTWRITE	3009	File is not opened for write operations.	You tried to update a file that has not been opened for write operations.
ESTDIN	3010	The stdin stream cannot be opened.	
ESTDOUT	3011	The stdout stream cannot be opened.	
ESTDERR	3012	The stderr stream cannot be opened.	
EBADSEEK	3013	The positioning parameter in fseek is not correct.	
EBADNAME	3014	The object name specified is not correct.	
EBADMODE	3015	The type variable specified on the open function is not correct.	The mode that you attempted to open the file in is not correct.
EBADPOS	3017	The position specifier is not correct.	
ENOPOS	3018	There is no record at the specified position.	You attempted to position to a record that does not exist in the file.

Name	Value	Text	Details
ENUMMBRS	3019	Attempted to use ftell on multiple members.	Remove all but one member from the file.
ENUMRECS	3020	The current record position is too long for ftell.	
EINVAL	3021	The value specified for the argument is not correct.	A function was passed incorrect argument values, or an operation was attempted on an object and the operation specified is not supported for that type of object.
EBADFUNC	3022	Function parameter in the signal function is not set.	
ENOENT	3025	No such path or directory.	The directory or a component of the path name specified does not exist.
ENOREC	3026	Record is not found.	
EPERM	3027	The operation is not permitted.	You must have appropriate privileges or be the owner of the object or other resource to do the requested operation.
EBADDATA	3028	Message data is not valid.	The message data that was specified for the error text is not correct.
EBUSY	3029	Resource busy.	An attempt was made to use a system resource that is not available at this time.
EBADOPT	3040	Option specified is not valid.	
ENOTUPD	3041	File is not opened for update operations.	
ENOTDLT	3042	File is not opened for delete operations.	
EPAD	3043	The number of characters written is shorter than the expected record length.	The length of the record is longer than the buffer size that was specified. The data written was padded to the length of the record.
EBADKEYLN	3044	A length that was not valid was specified for the key.	You attempted a record I/O against a keyed file. The key length that was specified is not correct.
EPUTANDGET	3080	A read operation should not immediately follow a write operation.	
EGETANDPUT	3081	A write operation should not immediately follow a read operation.	
EIOERROR	3101	A nonrecoverable I/O error occurred.	
EIORECERR	3102	A recoverable I/O error occurred.	
EACCES	3401	Permission denied.	An attempt was made to access an object in a way forbidden by its object access permissions.
ENOTDIR	3403	Not a directory.	A component of the specified path name existed, but it was not a directory when a directory was expected.

Name	Value	Text	Details
ENOSPC	3404	No space is available.	The requested operations required additional space on the device and there is no space left. This could also be caused by exceeding the user profile storage limit when creating or transferring ownership of an object.
EXDEV	3405	Improper link.	A link to a file on another file system was attempted.
EAGAIN	3406	Operation would have caused the process to be suspended.	
EWOULDBLOCK	3406	Operation would have caused the process to be suspended.	
EINTR	3407	Interrupted function call.	
EFAULT	3408	The address used for an argument was not correct.	In attempting to use an argument in a call, the system detected an address that is not valid.
ETIME	3409	Operation timed out.	
ENXIO	3415	No such device or address.	
EAPAR	3418	Possible APAR condition or hardware failure.	
ERECURSE	3419	Recursive attempt rejected.	
EADDRINUSE	3420	Address already in use.	
EADDRNOTAVAIL	3421	Address is not available.	
EAFNOSUPPORT	3422	The type of socket is not supported in this protocol family.	
EALREADY	3423	Operation is already in progress.	
ECONNABORTED	3424	Connection ended abnormally.	
ECONNREFUSED	3425	A remote host refused an attempted connect operation.	
ECONNRESET	3426	A connection with a remote socket was reset by that socket.	
EDESTADDRREQ	3427	Operation requires destination address.	
EHOSTDOWN	3428	A remote host is not available.	
EHOSTUNREACH	3429	A route to the remote host is not available.	
EINPROGRESS	3430	Operation in progress.	
EISCONN	3431	A connection has already been established.	
EMSGSIZE	3432	Message size is out of range.	
ENETDOWN	3433	The network currently is not available.	
ENETRESET	3434	A socket is connected to a host that is no longer available.	
ENETUNREACH	3435	Cannot reach the destination network.	

Name	Value	Text	Details
ENOBUFS	3436	There is not enough buffer space for the requested operation.	
ENOPROTOOPT	3437	The protocol does not support the specified option.	
ENOTCONN	3438	Requested operation requires a connection.	
ENOTSOCK	3439	The specified descriptor does not reference a socket.	
ENOTSUP	3440	Operation is not supported.	The operation, though supported in general, is not supported for the requested object or the requested arguments.
EOPNOTSUPP	3440	Operation is not supported.	The operation, though supported in general, is not supported for the requested object or the requested arguments.
EPFNOSUPPORT	3441	The socket protocol family is not supported.	
EPROTONOSUPPORT	3442	No protocol of the specified type and domain exists.	
EPROTOTYPE	3443	The socket type or protocols are not compatible.	
ERCVDERR	3444	An error indication was sent by the peer program.	
ESHUTDOWN	3445	Cannot send data after a shutdown.	
ESOCKTNOSUPPORT	3446	The specified socket type is not supported.	
ETIMEDOUT	3447	A remote host did not respond within the timeout period.	
EUNATCH	3448	The protocol required to support the specified address family is not available at this time.	
EBADF	3450	Descriptor is not valid.	A file descriptor argument was out of range, referred to a file that was not open, or a read or write request was made to a file that is not open for that operation.
EMFILE	3452	Too many open files for this process.	An attempt was made to open more files than allowed by the value of OPEN_MAX. The value of OPEN_MAX can be retrieved using the sysconf() function.
ENFILE	3453	Too many open files in the system.	A system limit has been reached for the number of files that are allowed to be concurrently open in the system.
EPIPE	3455	Broken pipe.	
ECANCEL	3456	Operation cancelled.	
EEXIST	3457	Object exists.	The object specified already exists and the specified operation requires that it not exist.

Name	Value	Text	Details
EDEADLK	3459	Resource deadlock avoided.	An attempt was made to lock a system resource that would have resulted in a deadlock situation. The lock was not obtained.
ENOMEM	3460	Storage allocation request failed.	A function needed to allocate storage, but no storage is available.
EOWNERTERM	3462	The synchronization object no longer exists because the owner is no longer running.	The process that had locked the mutex is no longer running, so the mutex was deleted.
EDESTROYED	3463	The synchronization object was destroyed, or the object no longer exists.	
ETERM	3464	Operation was terminated.	
ENOENT1	3465	No such file or directory.	A component of a specified path name did not exist, or the path name was an empty string.
ENOEQFLOG	3466	Object is already linked to a dead directory.	The link as a dead option was specified, but the object is already marked as dead. Only one dead link is allowed for an object.
EEMPTYDIR	3467	Directory is empty.	A directory with entries of only dot and dot-dot was supplied when a nonempty directory was expected.
EMLINK	3468	Maximum link count for a file was exceeded.	An attempt was made to have the link count of a single file exceed LINK_MAX. The value of LINK_MAX can be determined using the pathconf() or the fpathconf() function.
ESPIPE	3469	Seek request is not supported for object.	A seek request was specified for an object that does not support seeking.
ENOSYS	3470	Function not implemented.	An attempt was made to use a function that is not available in this implementation for any object or any arguments.
EISDIR	3471	Specified target is a directory.	The path specified named a directory where a file or object name was expected.
EROFS	3472	Read-only file system.	You have attempted an update operation in a file system that only supports read operations.
EUNKNOWN	3474	Unknown system state.	The operation failed because of an unknown system state. See any messages in the job log and correct any errors that are indicated, then retry the operation.
EITERBAD	3475	Iterator is not valid.	
EITERSTE	3476	Iterator is in wrong state for operation.	
EHRICLSBAD	3477	HRI class is not valid.	
EHRICLBAD	3478	HRI subclass is not valid.	
EHRITYPBAD	3479	HRI type is not valid.	

Name	Value	Text	Details
ENOTAPPL	3480	Data requested is not applicable.	
EHRIREQTYP	3481	HRI request type is not valid.	
EHRINAMEBAD	3482	HRI resource name is not valid.	
EDAMAGE	3484	A damaged object was encountered.	
ELOOP	3485	A loop exists in the symbolic links.	This error is issued if the number of symbolic links encountered is more than POSIX_SYMLOOP (defined in the limits.h header file). Symbolic links are encountered during resolution of the directory or path name.
ENAMETOOLONG	3486	A path name is too long.	A path name is longer than PATH_MAX characters or some component of the name is longer than NAME_MAX characters while _POSIX_NO_TRUNC is in effect. For symbolic links, the length of the name string substituted for a symbolic link exceeds PATH_MAX. The PATH_MAX and NAME_MAX values can be determined using the **pathconf()** function.
ENOLCK	3487	No locks are available.	A system-imposed limit on the number of simultaneous file and record locks was reached, and no more were available at that time.
ENOTEMPTY	3488	Directory is not empty.	You tried to remove a directory that is not empty. A directory cannot contain objects when it is being removed.
ENOSYSRSC	3489	System resources are not available.	
ECONVERT	3490	Conversion error.	One or more characters could not be converted from the source CCSID to the target CCSID.
E2BIG	3491	Argument list is too long.	
EILSEQ	3492	Conversion stopped due to input character that does not belong to the input codeset.	
ETYPE	3493	Object type mismatch.	The type of the object referenced by a descriptor does not match the type specified on the interface.
EBADDIR	3494	Attempted to reference a directory that was not found or was destroyed.	
EBADOBJ	3495	Attempted to reference an object that was not found, was destroyed, or was damaged.	
EIDXINVAL	3496	Data space index used as a directory is not valid.	
ESOFTDAMAGE	3497	Object has soft damage.	
ENOTENROLL	3498	User is not enrolled in system distribution directory.	You attempted to use a function that requires you to be enrolled in the system distribution directory and you are not.

Name	Value	Text	Details
EOFFLINE	3499	Object is suspended.	You have attempted to use an object that has had its data saved and the storage associated with it freed. An attempt to retrieve the object's data failed. The object's data cannot be used until it is successfully restored. The object's data was saved and freed either by saving the object with the STG(*FREE) parameter, or by calling an API.
EROOBJ	3500	Object is read-only.	You have attempted to update an object that can be read only.
EEAHDDSI	3501	Hard damage on extended attribute data space index.	
EEASDDSI	3502	Soft damage on extended attribute data space index.	
EEAHDDS	3503	Hard damage on extended attribute data space.	
EEASDDS	3504	Soft damage on extended attribute data space.	
EEADUPRC	3505	Duplicate extended attribute record.	
ELOCKED	3506	Area being read from or written to is locked.	The read or write of an area conflicts with a lock held by another process.
EFBIG	3507	Object too large.	The size of the object would exceed the system allowed maximum size.
EIDRM	3509	The semaphore, shared memory, or message queue identifier is removed from the system.	
ENOMSG	3510	The queue does not contain a message of the desired type and (msgflg logically ANDed with IPC_NOWAIT).	
EFILECVT	3511	File ID conversion of a directory failed.	» To recover from this error, run the Reclaim Storage (RCLSTG) command as soon as possible. «
EBADFID	3512	A file ID could not be assigned when linking an object to a directory.	The file ID table is missing or damaged. » To recover from this error, run the Reclaim Storage (RCLSTG) command as soon as possible. «
ESTALE	3513	File or object handle rejected by server.	
ESRCH	3515	No such process.	
ENOTSIGINIT	3516	Process is not enabled for signals.	An attempt was made to call a signal function under one of the following conditions: • The signal function is being called for a process that is not enabled for asynchronous signals. • The signal function is being called when the system signal controls have not been initialized.
ECHILD	3517	No child process.	

Name	Value	Text	Details
EBADH	3520	Handle is not valid.	
ETOOMANYREFS	3523	The operation would have exceeded the maximum number of references allowed for a descriptor.	
ENOTSAFE	3524	Function is not allowed.	Function is not allowed in a job that is running with multiple threads.
EOVERFLOW	3525	Object is too large to process.	The object's data size exceeds the limit allowed by this function.
EJRNDAMAGE	3526	Journal is damaged.	A journal or all of the journal's attached journal receivers are damaged, or the journal sequence number has exceeded the maximum value allowed. This error occurs during operations that were attempting to send an entry to the journal.
EJRNINACTIVE	3527	Journal is inactive.	The journaling state for the journal is *INACTIVE. This error occurs during operations that were attempting to send an entry to the journal.
EJRNRCVSPC	3528	Journal space or system storage error.	The attached journal receiver does not have space for the entry because the storage limit has been exceeded for the system, the object, the user profile, or the group profile. This error occurs during operations that were attempting to send an entry to the journal.
EJRNRMT	3529	Journal is remote.	The journal is a remote journal. Journal entries cannot be sent to a remote journal. This error occurs during operations that were attempting to send an entry to the journal.
ENEWJRNRCV	3530	New journal receiver is needed.	A new journal receiver must be attached to the journal before entries can be journaled. This error occurs during operations that were attempting to send an entry to the journal.
ENEWJRN	3531	New journal is needed.	The journal was not completely created, or an attempt to delete it did not complete successfully. This error occurs during operations that were attempting to start or end journaling, or were attempting to send an entry to the journal.
EJOURNALED	3532	Object already journaled.	A start journaling operation was attempted on an object that is already being journaled.
EJRNENTTOOLONG	3533	Entry is too large to send.	The journal entry generated by this operation is too large to send to the journal.
EDATALINK	3534	Object is a datalink object.	

Name	Value	Text	Details
ENOTAVAIL	3535	Independent Auxiliary Storage Pool (ASP) is not available.	The independent ASP is in Vary Configuration (VRYCFG) or Reclaim Storage (RCLSTG) processing. To recover from this error, wait until processing has completed for the independent ASP.
ENOTTY	3536	I/O control operation is not appropriate.	
EFBIG2	3540	Attempt to write or truncate file past its sort file size limit.	
ETXTBSY	3543	Text file busy.	» An attempt was made to execute an i5/OS PASE program that is currently open for writing, or an attempt has been made to open for writing an i5/OS PASE program that is being executed. «
EASPGRPNOTSET	3544	ASP group not set for thread.	
ERESTART	3545	A system call was interrupted and may be restarted.	
ESCANFAILURE	3546	Object had scan failure.	An object has been marked as a scan failure due to processing by an exit program associated with the scan-related integrated file system exit points.

Top | UNIX-Type APIs | APIs by category

Appendix. Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

```
IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY  10504-1785
U.S.A.
```

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

```
IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan
```

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Software Interoperability Coordinator, Department YBWA
3605 Highway 52 N
Rochester, MN 55901
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, IBM License Agreement for Machine Code, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Programming Interface Information

This Application Programming Interfaces (API) publication documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM i5/OS.

Trademarks

The following terms are trademarks of International Business Machines Corporation in the United States, other countries, or both:
Advanced 36
Advanced Function Printing
Advanced Peer-to-Peer Networking
AFP
AIX
AS/400
COBOL/400
CUA
DB2
DB2 Universal Database
Distributed Relational Database Architecture
Domino
DPI
DRDA
eServer
GDDM
IBM
Integrated Language Environment
Intelligent Printer Data Stream
IPDS
i5/OS
iSeries
Lotus Notes
MVS
Netfinity
Net.Data
NetView
Notes
OfficeVision
Operating System/2
Operating System/400
OS/2
OS/400
PartnerWorld
PowerPC
PrintManager
Print Services Facility
RISC System/6000
RPG/400
RS/6000
SAA
SecureWay
System/36
System/370
System/38
System/390
VisualAge
WebSphere
xSeries

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

Terms and Conditions

Permissions for the use of these Publications is granted subject to the following terms and conditions.

Personal Use: You may reproduce these Publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative works of these Publications, or any portion thereof, without the express consent of IBM.

Commercial Use: You may reproduce, distribute and display these Publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these Publications, or reproduce, distribute or display these Publications or any portion thereof outside your enterprise, without the express consent of IBM.

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the Publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the Publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations. IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE

Printed in USA