System.IO.Directory Class

```
[ILAsm]
.class public sealed Directory extends System.Object

[C#]
public sealed class Directory
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

Summary

Provides information and performs operations on directories.

Inherits From: System.Object

Library: BCL

Thread Safety: All public static members of this type are safe for multithreaded operations. No instance members are guaranteed to be thread safe.

Description

Implementations are required to preserve the case of file and directory path strings, and to be case sensitive if and only if the current platform is case-sensitive.

[*Note:* In most `Directory` methods that accept *path* arguments, the path can refer to a file or a directory.]

Directory.Delete(System.String, System.Boolean) Method

```
[ILAsm]
.method public hidebysig static void Delete(string path, bool
recursive)

[C#]
public static void Delete(string path, bool recursive)
```

Summary

Deletes the specified directory and, if indicated, any subdirectories in the directory.

Parameters

Parameter	Description
path	A `System.String` containing the name of the directory to delete. This directory must be writable and cannot contain files unless *recursive* is true.
recursive	Specify `true` to delete subdirectories and files in *path*; otherwise, specify `false`.

Description

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters.
System.ArgumentNullException	*path* is `null`.
System.IO.DirectoryNotFoundException	The specified *path* was not found.
System.IO.IOException	The directory specified by *path* is read-only, or *recursive* is `false` and *path* is not an empty directory.

System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to write to the specified directory. See `System.Security.Permissions.FileIOPermissionAccess.Write`.

Directory.Delete(System.String) Method

```
[ILAsm]
.method public hidebysig static void Delete(string path)

[C#]
public static void Delete(string path)
```

Summary

Deletes the empty directory specified in *path*.

Parameters

Parameter	Description
path	A `System.String` containing the name of the directory to delete. This directory must be writable and empty.

Description

This method behaves identically to `System.IO.Directory.Delete`(*path*, `false`).

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters.
System.ArgumentNullException	*path* is `null`.
System.IO.DirectoryNotFoundException	The specified *path* was not found.
System.IO.IOException	The directory specified by *path* is read-only or is not empty.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required

	permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to write to the specified directory. See `System.Security.Permissions.FileIOPermissionAccess.Write`.

Directory.Exists(System.String) Method

```
[ILAsm]
.method public hidebysig static bool Exists(string path)

[C#]
public static bool Exists(string path)
```

Summary

Returns a `System.Boolean` indicating whether the specified directory exists.

Parameters

Parameter	Description
path	A `System.String` containing the name of the directory to check.

Return Value

`true` if the caller has the required permissions and *path* contains the name of an existing directory; otherwise, `false`. If *path* is `null`, a zero-length string, or contains the name of a file, returns `false`.

Description

If the caller does not have sufficient permissions to read the files in the directory specified by *path*, no exception is thrown and the method returns `false` regardless of the existence of *path*.

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to read the specified directory. See `System.Security.Permissions.FileIOPermissionAccess.Read`.

Directory.GetCreationTime(System.String) Method

```
[ILAsm]
.method public hidebysig static valuetype System.DateTime
GetCreationTime(string path)

[C#]
public static DateTime GetCreationTime(string path)
```

Summary

Returns the creation date and time of the specified file or directory.

Parameters

Parameter	Description
path	A `System.String` containing the name of the file or directory for which to obtain creation date and time information.

Return Value

A `System.DateTime` structure set to the creation date and time for the specified directory. This value is expressed in local time.

Platforms that do not support this feature return `System.DateTime.MinValue`.

Description

This method is equivalent to `System.IO.File.GetCreationTime` (*path*).

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more

	implementation-specific invalid characters.
System.ArgumentNullException	*path* is `null`.
System.IO.IOException	The directory specified by *path* was not found.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to read the specified file or directory. See `System.Security.Permissions.FileIOPermissionAccess.Read`.

Directory.GetCurrentDirectory() Method

```
[ILAsm]
.method public hidebysig static string GetCurrentDirectory()

[C#]
public static string GetCurrentDirectory()
```

Summary

Returns the application's current working directory.

Return Value

A `System.String` containing the path of the current working directory.

Platforms that do not support this feature return `System.String.Empty`.

Exceptions

Exception	Condition
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to access path information for the current directory. See `System.Security.Permissions.FileIOPermissionAccess.PathDiscovery`

Directory.GetDirectories(System.String) Method

```
[ILAsm]
.method public hidebysig static string[] GetDirectories(string path)

[C#]
public static string[] GetDirectories(string path)
```

Summary

Returns the names of subdirectories in the specified directory.

Parameters

Parameter	Description
path	A `System.String` containing the name of the directory for which an array of subdirectory names is returned.

Return Value

A `System.String` array containing the names of subdirectories in *path*.

Description

This method is identical to `System.IO.Directory.GetDirectories` (*path*, "*").

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

The names returned by this method are prefixed with the directory information provided in *path*.

Exceptions

Exception	Condition
System.ArgumentNullException	*path* is `null`.
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains implementation-specific invalid

	characters.
System.IO.DirectoryNotFoundException	*path* was not found.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.IO.IOException	*path* is a file name.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to access path information for the specified directory and its subdirectories. See `System.Security.Permissions.FileIOPermissionAccess.PathDiscovery`.

Directory.GetDirectories(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig static string[] GetDirectories(string path,
string searchPattern)

[C#]
public static string[] GetDirectories(string path, string
searchPattern)
```

Summary

Returns the names of subdirectories in the specified directory that match the specified search pattern.

Parameters

Parameter	Description
path	A `System.String` containing the starting location for the search.
searchPattern	A `System.String` containing the text pattern to match against the names of subdirectories of *path*. *searchPattern* cannot end with "..", or contain ".." followed by `System.IO.Path.DirectorySeparatorChar` or `System.IO.Path.AltDirectorySeparatorChar`.

Return Value

A `String` array containing the names of subdirectories of *path* that match *searchPattern*.

Description

The following wild card specifiers are permitted in *searchPattern*:

Wild card	Description
*	Zero or more characters.
?	Exactly one character.

The period (".") character, if immediately followed by a wild card specifier, indicates that the period or the empty string matches the pattern. For example, "foo.*" and "foo.?" match "foo". Note that "foo.*" and "foo*" behave identically. If the period is not immediately followed by a wildcard, it has no special meaning (it represents a period).

Characters other than the wild card specifiers represent themselves, for example, the *searchPattern* string "*t" searches for all names in*path* ending with the letter "t". The *searchPattern* string "s*" searches for all names in *path* beginning with the letter "s".

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentNullException	*path* or *searchPattern* is `null`.
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains implementation-specific invalid characters. *searchPattern* does not contain a valid pattern.
System.IO.DirectoryNotFoundException	*path* was not found.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.IO.IOException	*path* is a file name.
System.UnauthorizedAccessException	The caller does not have permission to access the requested information.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to access path information for the specified directory and its subdirectories. See `System.Security.Permissions.FileIOPermissionAccess.PathDiscovery`.

Directory.GetDirectoryRoot(System.String) Method

```
[ILAsm]
.method public hidebysig static string GetDirectoryRoot(string path)

[C#]
public static string GetDirectoryRoot(string path)
```

Summary

Returns the path root component of the specified path.

Parameters

Parameter	Description
path	A `System.String` containing the name of a file or directory.

Return Value

A `System.String` containing the root information for the specified path.

Platforms that do not support this feature return `System.String.Empty`.

Description

This method obtains the full path information for *path*, as returned by `System.IO.Path.GetFullPath` (*path*) and returns the path root component. The specified path is not required to exist.

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more implementation-specific invalid

	characters.
System.ArgumentNullException	*path* is `null`.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Example

The following example demonstrates the `System.IO.Directory.GetDirectoryRoot` method.

[C#]

```
using System;
using System.IO;
class GetDirectoryTest {
  public static void Main() {
    string [] paths = {

@"\ecmatest\examples\pathtests.txt",
      "pathtests.xyzzy",
      @"\",
      @"C:\",
      @"\\myserver\myshare\foo\bar\baz.txt"
    };
    foreach (string pathString in paths) {
      string s = Directory.GetDirectoryRoot(pathString);
      Console.WriteLine("Path: {0} Directory Root is {1}",pathString,
s== null? "null":s);
    }
  }
}
```

The output is

```
Path: \ecmatest\examples\pathtests.txt Directory Root is C:\


Path: pathtests.xyzzy Directory Root is C:\


Path: \ Directory Root is C:\


Path: C:\ Directory Root is C:\


Path: \\myserver\myshare\foo\bar\baz.txt Directory Root is
\\myserver\myshare
```

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to access path information for the specified file or directory. See `System.Security.Permissions.FileIOPermissionAccess.PathDiscovery`

Directory.GetFiles(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig static string[] GetFiles(string path,
string searchPattern)

[C#]
public static string[] GetFiles(string path, string searchPattern)
```

Summary

Returns the names of files in the specified directory that match the specified search pattern.

Parameters

Parameter	Description
path	A `System.String` containing the name of the directory to search.
searchPattern	A `System.String` containing the text pattern to match against the names of files in *path*. *searchPattern* cannot end with "..", or contain ".." followed by `System.IO.Path.DirectorySeparatorChar` or `System.IO.Path.AltDirectorySeparatorChar`.

Return Value

A `System.String` array containing the names of files in the specified directory that match the specified search pattern.

Description

The following wild card specifiers are permitted in *searchPattern*:

Wild card	Description
*	Zero or more characters.
?	Exactly one character.

The period (".") character, if immediately followed by a wild card specifier, indicates that the period or the empty string matches the pattern. For example, "foo.*" and "foo.?" match "foo". Note that "foo.*" and "foo*" behave identically. If the period is not immediately followed by a wildcard, it has no special meaning (it represents a period).

Characters other than the wild card specifiers and the period always represent themselves, for example, the *searchPattern* string "*t" searches for all names in *path* ending with the letter "t". The *searchPattern* string "s*" searches for all names in *path* beginning with the letter "s".

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentNullException	*searchPattern* or *path* is `null`.
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters. -or- *searchPattern* does not contain a valid pattern.
System.IO.IOException	*path* is an existing file name.
System.IO.DirectoryNotFoundException	*path* was not found.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to access path information for the specified directory and the files in that directory. See `System.Security.Permissions.FileIOPermissionAccess.PathDiscovery`

Directory.GetFiles(System.String) Method

```
[ILAsm]
.method public hidebysig static string[] GetFiles(string path)

[C#]
public static string[] GetFiles(string path)
```

Summary

Returns the names of all files in the specified directory.

Parameters

Parameter	Description
path	A `System.String` containing the name of the directory for which file names are returned.

Return Value

A `System.String` array containing the names of the files in the specified directory.

Platforms that do not support this feature return `null`.

Description

This method is identical to `System.IO.Directory.GetFiles` (*path*, "*").

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentNullException	*path* is null.
System.ArgumentException	path is a zero-length string, contains only white space, or contains one or more implementation-specific invalid

	characters.
System.IO.DirectoryNotFoundException	*path* was not found.
System.IO.IOException	*path* is a file name.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to access path information for the specified directory and the files in that directory. See `System.Security.Permissions.FileIOPermissionAccess.PathDiscovery`.

Directory.GetFileSystemEntries(System.String) Method

```
[ILAsm]
.method public hidebysig static string[] GetFileSystemEntries(string path)

[C#]
public static string[] GetFileSystemEntries(string path)
```

Summary

Returns the names of all files and subdirectories in the specified directory.

Parameters

Parameter	Description
path	A `System.String` containing the name of the directory for which file and subdirectory names are returned.

Return Value

A `System.String` array containing the names of file system entries in the specified directory.

Description

This method is identical to `System.IO.Directory.GetFileSystemEntries` (*path*, "*").

The names returned by this method are prefixed with the directory information provided in *path*. The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentNullException	*path* is `null`.
System.ArgumentException	*path* is a zero-length string, contains

	only white space, or contains one or more implementation-specific invalid characters.
System.IO.DirectoryNotFoundException	*path* was not found.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.IO.IOException	*path* is a file name.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to access path information for the specified directory. See `System.Security.Permissions.FileIOPermissionAccess.PathDiscovery`

Directory.GetFileSystemEntries(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig static string[] GetFileSystemEntries(string path, string searchPattern)

[C#]
public static string[] GetFileSystemEntries(string path, string searchPattern)
```

Summary

Returns an array of file and directory names matching the specified search criteria.

Parameters

Parameter	Description
path	A `System.String` containing the name of the directory to search.
searchPattern	A `System.String` containing the text pattern for which to search. *searchPattern* cannot end with "..", or contain ".." followed by `System.IO.Path.DirectorySeparatorChar` or `System.IO.Path.AltDirectorySeparatorChar`.

Return Value

A `String` array containing file and directory names matching the specified search criteria.

Description

The following wild card specifiers are permitted in *searchPattern*:

Wild card	Description
*	Zero or more characters.
?	Exactly one character.

The period (".") character, if immediately followed by a wild card specifier, indicates that the period or the empty string matches the pattern. For example, "foo.*" and "foo.?" match "foo". Note that "foo.*" and "foo*" behave identically. If the period is not immediately followed by a wildcard, it has no special meaning

(it represents a period).

Characters other than the wild card specifiers represent themselves, for example, the *searchPattern* string "*t" searches for all names in *path* ending with the letter "t". The *searchPattern* string "s*" searches for all names in *path* beginning with the letter "s".

The names returned by this method are prefixed with the directory information provided in *path*. The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentNullException	*searchPattern* or *path* is `null`.
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters. -or- *searchPattern* does not contain a valid pattern.
System.IO.DirectoryNotFoundException	*path* was not found.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.IO.IOException	*path* is a file name.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to access path information for the specified directory. See `System.Security.Permissions.FileIOPermissionAccess.PathDiscovery`

Directory.GetLastAccessTime(System.String) Method

```
[ILAsm]
.method public hidebysig static valuetype System.DateTime
GetLastAccessTime(string path)

[C#]
public static DateTime GetLastAccessTime(string path)
```

Summary

Returns the date and time the specified file or directory was last accessed.

Parameters

Parameter	Description
path	A `System.String` containing the name of the file or directory for which to obtain access date and time information.

Return Value

A `System.DateTime` structure set to the date and time the specified file or directory was last accessed. This value is expressed in local time.

Platforms that do not support this feature return `System.DateTime.MinValue`.

Description

This method is equivalent to `System.IO.File.GetLastAccessTime` (*path*).

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more

	implementation-specific invalid characters.
System.ArgumentNullException	*path* is `null`.
System.IO.IOException	The specified path was not found.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to read the specified file or directory. See `System.Security.Permissions.FileIOPermissionAccess.Read`.

Directory.GetLastWriteTime(System.String) Method

```
[ILAsm]
.method public hidebysig static valuetype System.DateTime
GetLastWriteTime(string path)

[C#]
public static DateTime GetLastWriteTime(string path)
```

Summary

Returns the date and time the specified file or directory was last written to.

Parameters

Parameter	Description
path	A `System.String` containing the name of the file or directory for which to obtain modification date and time information.

Return Value

A `System.DateTime` structure set to the date and time the specified file or directory was last written to. This value is expressed in local time.

Platforms that do not support this feature return `System.DateTime.MinValue`.

Description

This method is equivalent to `System.IO.File.GetLastWriteTime` (*path*).

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more

	implementation-specific invalid characters.
System.ArgumentNullException	*path* is `null`.
System.IO.IOException	The specified path was not found.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to read the specified file or directory. See `System.Security.Permissions.FileIOPermissionAccess.Read`.

Directory.Move(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig static void Move(string sourceDirName,
string destDirName)

[C#]
public static void Move(string sourceDirName, string destDirName)
```

Summary

Moves a file or a directory and its contents to a new location.

Parameters

Parameter	Description
sourceDirName	A `System.String` containing the name of the file or directory to move.
destDirName	A `System.String` containing the new location for *sourceDirName*. This string cannot identify an existing file or directory.

Description

The *destDirName* argument cannot specify a location on a different disk or volume than *sourceDirName*. The *sourceDirName* and *destDirName* arguments cannot identify the same file or directory.

[*Note:* This method throws a `System.IO.IOException` if, for example, you try to move "\mydir" to "\public", and "\public" already exists. You must specify "\public\mydir" as the *destDirName*.]

The *sourceDirName* and *destDirName* arguments are permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

Exceptions

Exception	Condition
System.ArgumentException	*sourceDirName* or *destDirName* is a

	zero-length string, contains only white space, or contains implementation-specific invalid characters.
System.ArgumentNullException	*sourceDirName* or *destDirName* is `null`.
System.UnauthorizedAccessException	The caller does not have the required permission.
System.IO.IOException	An attempt was made to move a directory to a different volume, -or- *destDirName* already exists. -or- *sourceDirName* and *destDirName* refer to the same file or directory.
System.IO.DirectoryNotFoundException	*sourceDirName* was not found.
System.IO.PathTooLongException	The length or absolute path information for *sourceDirName* or *destDirName* exceeds the system-defined maximum length.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to read from *sourceDirName*, and write to *sourceDirName* and *destDirName*. See `System.Security.Permissions.FileIOPermissionAccess.Read`, `System.Security.Permissions.FileIOPermissionAccess.Write`.

Directory.SetCreationTime(System.String, System.DateTime) Method

```
[ILAsm]
.method public hidebysig static void SetCreationTime(string path,
valuetype System.DateTime creationTime)

[C#]
public static void SetCreationTime(string path, DateTime
creationTime)
```

Summary

Sets the creation date and time for the specified file or directory.

Parameters

Parameter	Description
path	A `System.String` containing the name of the file or directory for which to set the creation date and time information.
creationTime	A `System.DateTime` containing the value to set for the creation date and time of *path*. This value is expressed in local time.

Description

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

On platforms that do not support this feature, this method has no effect. If this feature is supported, the range of dates that is valid for this operation is implementation-specific.

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters.
System.ArgumentOutOfRangeException	*creationTime* specifies a value outside the range of date/times permitted for

	this operation.
System.ArgumentNullException	*path* is `null`.
System.IO.FileNotFoundException	*path* was not found.
System.IO.IOException	An I/O error occurred while performing the operation.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to write to the specified file or directory. See `System.Security.Permissions.FileIOPermissionAccess.Write`.

Directory.SetCurrentDirectory(System.String) Method

```
[ILAsm]
.method public hidebysig static void SetCurrentDirectory(string
path)

[C#]
public static void SetCurrentDirectory(string path)
```

Summary

Sets the application's current working directory to the specified directory.

Parameters

Parameter	Description
path	A `System.String` containing the path to which the current working directory is set.

Description

When the application terminates, the working directory is restored to its original location (the directory where the process was started).

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

On platforms that do not support this feature, this method has no effect.

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters.
System.ArgumentNullException	*path* is `null`.
System.IO.FileNotFoundException	*path* was not found.
System.IO.IOException	An I/O error occurred while performing the

	operation.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.Security.SecurityException	The caller does not have the required permission to access unmanaged code.

Directory.SetLastAccessTime(System.String, System.DateTime) Method

```
[ILAsm]
.method public hidebysig static void SetLastAccessTime(string path,
valuetype System.DateTime lastAccessTime)

[C#]
public static void SetLastAccessTime(string path, DateTime
lastAccessTime)
```

Summary

Sets the date and time the specified file or directory was last accessed.

Parameters

Parameter	Description
path	A `System.String` containing the name of the file or directory for which to set the access date and time information.
lastAccessTime	A `System.DateTime` containing the value to set for the access date and time of *path*. This value is expressed in local time.

Description

The *path* argument is permitted to specify relative or absolute path information. Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

On platforms that do not support this feature, this method has no effect. If this feature is supported, the range of dates that is valid for this operation is implementation-specific.

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters.
System.ArgumentNullException	*path* is `null`.

System.ArgumentOutOfRangeException	*lastAccessTime* specifies a value outside the range of date/times permitted for this operation.
System.IO.FileNotFoundException	*path* was not found.
System.IO.IOException	An I/O error occurred while performing the operation.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to write to the specified file or directory. See `System.Security.Permissions.FileIOPermissionAccess.Write`.

Directory.SetLastWriteTime(System.String, System.DateTime) Method

```
[ILAsm]
.method public hidebysig static void SetLastWriteTime(string path,
valuetype System.DateTime lastWriteTime)

[C#]
public static void SetLastWriteTime(string path, DateTime
lastWriteTime)
```

Summary

Sets the date and time a directory was last written to.

Parameters

Parameter	Description
path	A `System.String` containing the name of the directory for which to set the date and time information.
lastWriteTime	A `System.DateTime` containing the value to set for the last write date and time of *path*. This value is expressed in local time.

Description

Relative path information is interpreted as relative to the current working directory. [*Note:* To obtain the current working directory, see `System.IO.Directory.GetCurrentDirectory`.]

On platforms that do not support this feature, this method has no effect. If this feature is supported, the range of dates that is valid for this operation is implementation-specific.

Exceptions

Exception	Condition
System.ArgumentException	*path* is a zero-length string, contains only white space, or contains one or more implementation-specific invalid characters.
System.ArgumentNullException	*path* is `null`.

System.ArgumentOutOfRangeException	*lastWriteTime* specifies a value outside the range of date/times permitted for this operation.
System.IO.FileNotFoundException	*path* was not found.
System.IO.IOException	An I/O error occurred while performing the operation.
System.IO.PathTooLongException	The length of *path* or the absolute path information for *path* exceeds the system-defined maximum length.
System.UnauthorizedAccessException	The caller does not have the required permission.

Permissions

Permission	Description
System.Security.Permissions.FileIOPermission	Requires permission to write to the specified file. See `System.Security.Permissions.FileIOPermissionAccess.Write`.