Article

Decompressing and extracting an archived directory

Recreate an entire file system directory from an archive file.

Overview

In this article, you'll learn how to use AppleArchive to decompress and extract a previously compressed file system directory.

The code below decompresses the file generated using the steps explained in Compressing file system directories and writes the files to a directory named `dest`.

Create the file stream to read the source archive

The `ArchiveByteStream` class provides static factory methods that create streams for different functions. In this case, use `fileStream(path:mode:options:permissions:)` to create a byte stream that reads the source file:

```
let archiveFilePath = FilePath(NSTemporaryDirectory() + "directory.aar")

guard let readFileStream = ArchiveByteStream.fileStream(
        path: archiveFilePath,
        mode: .readOnly,
        options: [ ],
        permissions: FilePermissions(rawValue: 0o644)) else {
    return
}
defer {
    try? readFileStream.close()
```

```
}
```

Create the decompression stream

Create the decompression stream. Specify the file-reading stream as the input stream that provides the compressed data:

```
guard let decompressStream = ArchiveByteStream.decompressionStream(readingFrom: read
    return
}
defer {
    try? decompressStream.close()
}
```

Create the decoding stream

Create a decoding stream that provides archive elements from the raw, decompressed data:

```
guard let decodeStream = ArchiveStream.decodeStream(readingFrom: decompressStream) e
    print("unable to create decode stream")
    return
}
defer {
    try? decodeStream.close()
}
```

Specify the destination

Specify a destination path for the decompressed directory contents. The following code checks that the destination directory exists and creates the destination if neccessary:

```
let decompressPath = NSTemporaryDirectory() + "dest/"

if !FileManager.default.fileExists(atPath: decompressPath) {
    do {
        try FileManager.default.createDirectory(atPath: decompressPath,
                                                withIntermediateDirectories: false)
    } catch {
        fatalError("Unable to create destination directory.")
    }
```

```
}

let decompressDestination = FilePath(decompressPath)
```

Create the extract stream

Create an extract stream that receives archive elements, and extracts to the specified directory:

```
guard let extractStream = ArchiveStream.extractStream(extractingTo: decompressDestir
                                                      flags: [ .ignoreOperationNotPe
    return
}
defer {
    try? extractStream.close()
}
```

Decompress and extract the archived directory

Finally, call `process(readingFrom:writingTo:)` to write the output of the decode stream to the extract stream. In turn, the extract stream extracts each archive element to the decompression destination:

```
do {
    _ = try ArchiveStream.process(readingFrom: decodeStream,
                                  writingTo: extractStream)
} catch {
    print("process failed")
}
```

On return, the operation recreates the contents of the directory previously archived in `directory.aar` in `NSTemporaryDirectory()` + `"dest/"`.

See Also

Directories, Files, and Data Archives

- Compressing single files

  Compress a single file and store the result on the file system.

- Decompressing single files
  Recreate a single file from a compressed file.

- Compressing file system directories
  Compress the contents of an entire directory and store the result on the file system.

- Compressing and saving a string to the file system
  Compress the contents of a Unicode string and store the result on the file system.

- Decompressing and Parsing an Archived String
  Recreate a string from an archive file.