clickclickmoon / **s3d_convert**



master Go to file Add file Code

**clickclickmoon** Comments clean... ... on Oct 11, 2009 15

| | | |
|---|---|---|
| README.tex... | Editing changes thanks to Kal... | 11 years ago |
| s3d.erl | Comments cleaned up | 11 years ago |

README.textile

# s3d_convert

> An erlang module for packing and unpacking s3d archive files.

This is a tool for packing and unpacking resource archives for the EverQuest
client in its .s3d file format. The code is contained in a single module
with a single exported function: unpack/1. The unpack/1 function takes a
filename of a .s3d file and creates a folder containing the unpacked archive data.

The credit for this application really goes to WindCatcher on the EQEmu forums.
He figured out all of the details on how these files work, and even provided an
application with sourcecode (S3DSpy) that would allow you to browse the archive
and look at the textures contained within.

## About

An erlang module for packing and unpacking s3d archive files.

Readme

## Releases

No releases published

## Packages

No packages published

## Languages

- **Erlang** 100.0%

S3DSpy was written in Delphi, which is not a language I'm particularly fond of
and have no intentions of using for other projects in the future. I wanted to
make a general module/library for the s3d format that I could use on the command
line or in a Makefile to unpack data easily. I also needed to learn the details
and I've always thought there is no better way to learn than to do. So I did.

Thank you again to WindWatcher, all the EQEmu developers and contributors, along
with the EQClassic project and it's collection of developers.

## s3d Package Format

There are only two data types used internally in the package: a single byte
that is used for the actual datablocks, and a four byte item (unsigned long
int) for checksums, counts, and offset pointers. The four byte integers are
stored as little endian.

### File Header

The initial three items can be thought of as the file header, which contains just a
little bit of data that seems to be consistant between files. The first of these
is a pointer to another location in the file, which is very important. The
other two are the constant header data that is the same between files.

- [4 bytes] @ 0×00 – **file_listing_offset**
- [4 bytes] @ 0×04 – **pfs_string**
- [4 bytes] @ 0×08 – **file_flags**

***file_listing_offset***
This is a pointer to the file meta block.

***pfs_string***
This is always 542328400. As a character byte string, it is 'PFS ' which is an
identifier as a PFS archive.

***file_flags***
The last item in the header is somewhat of an unknown. It does not appear to
change between files and is never referenced elsewhere in the file. Given that
the number comes out to 131072, I suspect that this is a file flag bit mask.
The truth may never be found, but for now we just look for the known good
number and go with that.

## Meta Listing Block

The meta listing block contains the checksums, data offsets, and actual file
size for all of the files contained in the archive. The first item in this
block is a count of the number of entries (aka. files) in the archive.

The meta data for each file is in a twelve byte block and contains a four byte
field for a file checksum, the pointer to the first data block, and the size of
the inflated file.

- [4 bytes] @ file_listing_offset – **EntryCount**
  EntryCount times in a continuous block
- [4 bytes] @ X + 0×00 – **checksum**
- [4 bytes] @ X + 0×04 – **offset**
- [4 bytes] @ X + 0×08 – **file_size**

***checksum***
IEEE 802.3 Ethernet CRC-32 checksum for the end file.

***offset***
Pointer to the first compressed data block for the file.

***file_size***
The real size of the decompressed file.

## File Data Blocks

Each file is made up of one or more data blocks. Each of these data blocks has
an eight byte header. The previously mentioned file meta offset points to
the first block, subsequent blocks will follow behind. This is done to ensure
the blocks can be zlib compressed/uncompressed.

- [4 bytes] @ meta offset + 0×00 – **DeflatedLength**
- [4 bytes] @ meta offset + 0×04 – **InflatedLength**
- [DeflatedLength bytes] @ meta offset + 0×08 – **Data Bytes**

***DeflatedLength***
The compressed size of this data block.

***InflatedLength***
The inflated size of this data block. Used to ensure we have collected all of
the data blocks for the file.

***Data Bytes***
The raw data block for this file (one of possibly many), zlib compressed.

## Directory File

The very last file (in terms of offset into the file) is zlib compressed,
and contains the list of file names in file offset order. The first four bytes
are unknown, then there are four bytes of separator data (0, 13, 0, 0), and then
the file name. This directory file is not listed inside of itself.

h.4 Data Footer

Sometimes the s3d archive has a small signature on the bottom. When there, it
reads “STEVEXXXX” where XXXX is the date. This footer is ignored since it is
not needed to parse and extract the file.

## Rough Drawing

(Not to scale)

```
 | file_listing_offset | pfs_string | file_flags |
 ------------------------------------------------
--
 | DeflatedLength | InflatedLength | DataBlocks[...]
|
 [ .... EntryCount of these blocks total ... ... ...
]
 | DeflatedLength | InflatedLength | DataBlocks[...]
| <-- Directory File
 ------------------------------------------------
--
 | EntryCount |
 | file_checksum | first_block_offset | file_size |
 [ ... EntryCount number of these blocks total .. ]
 ------------------------------------------------
--
 | STEVE | DATE |
```