UPS File Format Specification

April 18, 2008

Contents

1 Introduction

The goal of UPS is different from that of NINJA2, or xdelta. It is designed to be a direct replacement for IPS. The difficulty of implementing it is equal to that of IPS.

UPS supports any file size, though the binary is currently limited to 4GB files due to libc limitations. Because of a condensed pointer encoding method, the maximum file size is indefinite. UPS is completely future proof in that regard.

UPS is designed for patching files. It will not perform advanced clipping methods to move data around and restructure files like xdelta will. However, it is possible for anyone to implement UPS, whereas there has only ever been one xdelta implementation. The complexity of such implementations proves itself to be too difficult for easy implementation or integration. If these features are desired, the alternatives of xdelta and bsdiff already exist.

UPS doesn't compete with NINJA, they will complement each other. NINJA3 will use UPS internally as the raw patch data, and will handle detection and support for each individual system, as needed. NINJA3 behaves as a container, much like the relationship between OGG and Vorbis.

The reason UPS does not include compression is because ZIP, RAR, and 7z have and will always do it better. It is better to just have a larger patch handled by external compression. Most emulators support patches inside archives regardless. This furthers the key idea of easy implementation.

Finally, UPS is a finalized spec. Patches created will work with all future versions.

2 Advantages

- simple file format, easy for anyone to implement.
- automatic bi-directional patching. The same patch can both patch and unpatch a game.
- CRC32 checksums on the original, modified and patch files guarantees patches will not apply to the incorrect files
- infinite file sizes. No more 16MB limitation as with IPS.
- Windows / Linux GUI patchers, core library written in ISO C++9x.
- UPS is public domain

3 File Structure

3.1 Signature

- 4 bytes: "UPS1"

3.2 File Sizes

These are exact file sizes, variable length-encoded.

- Input file size
- Output file size

3.3 Patch Blocks

Blocks of changes are stored consecutively until EOF - 12 is reached.

- Relative Difference Offset: A variable length-encoded pointer describing the current difference between the current *input*, *output* file pointer and the next different byte.
- Input ^ Output: The XOR (exclusive or), of the differing input byte and the output byte. Data is stored as XOR to allow for bi-linear patching; to revert back to the original file with the same patch by applying it again.

- If reading past input file EOF, XOR with 0x00
- Terminating byte: "0x00"

3.4 Checksums

Values should be verified when applying the UPS patch. This ensures the integrity of the patch itself, and that the patch is being applied to the correct file.

- 4 byte Input file CRC32 checksum
- 4 byte Output file CRC32 checksum
- 4 byte Patch CRC32 checksum, excluding this checksum data.

3.5 Encoding Pointers

Pseudo code:

```
def encode(uint64_t offset) {
  loop {
    uint64_t x = offset bit-wise and 0x7f
    offset = offset right bit shift 7
    if(offset == 0) {
      zwrite(0x80 bit-wise or x);
      break;
    }
    zwrite(x);
    offset = offset - 1;
  }
}
```