ARC/DIR Format and `arcdir_tool` Complete Technical Documentation

February 1, 2026

Contents

1 Introduction

This document provides a structured technical description of the paired `.arc`/`.dir` archive format and the `arcdir_tool` utility used for extracting, packing, and managing archive data. It covers command-line syntax, internal algorithms, binary file structures, alignment rules, and practical examples.

The utility is implemented in C, ensuring:

- High performance through direct file I/O;
- Cross-platform compatibility (Windows and POSIX systems);
- Deterministic behavior for identical input data.

2 Concept Overview

The archive format consists of two interrelated files:

- `.arc` — a sequential concatenation of raw file data;
- `.dir` — an index containing metadata such as file offsets, sizes, and paths.

The `.dir` file functions as a structured index for `.arc`, mapping file paths to their offsets and sizes.

3 System Architecture

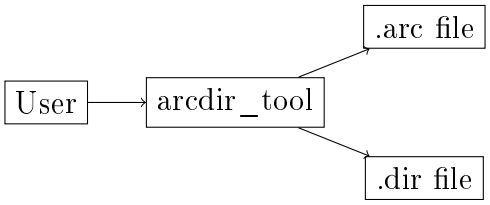

Table 1: Overall architecture of `arcdir_tool`

4 Program Execution Flow

The execution flow of `arcdir_tool` can be described as a linear sequence of operations:

1. Parse command-line arguments and identify the mode: `EXTRACT`, `PACK`, or `PACK_BIN`.
2. **EXTRACT mode:**
   - Open ARC and DIR files in read mode;
   - Read DIR entries sequentially;
   - Apply optional path redirection rules;
   - Write file data to the corresponding output paths.
3. **PACK / PACK_BIN modes:**
   - Recursively scan the specified directories and files;
   - Apply extension filter (`.bin` for PACK_BIN mode);
   - Sort all discovered paths lexicographically;
   - Write DIR entries and sequentially place file data in ARC;
   - Apply padding for path strings (4-byte alignment) and file data (32-byte alignment).
4. Close all files and terminate execution.

5 Command-Line Interface

5.1 Syntax

```
arcdir_tool EXTRACT <archive.arc> <index.dir> [<src> <dst>]...
arcdir_tool PACK <archive.arc> <index.dir> [path...]
arcdir_tool PACK_BIN <archive.arc> <index.dir> [path...]
```

5.2 Modes

5.2.1 EXTRACT

- Extracts files from an existing `.arc`/`.dir` archive.
- If no `<src> <dst>` pairs are provided, all files are extracted to their stored paths.
- If pairs are provided, only matching files are extracted to the specified destinations.

5.2.2 PACK

- Creates a new `.arc`/`.dir` archive pair.
- Arguments may be files or directories;
- Directories are traversed recursively;
- All discovered files are included in alphabetical order.

5.2.3 PACK_BIN

- Identical to PACK mode, but recursively discovered files are filtered by the extension `.bin`;
- Explicitly listed files are included regardless of extension.

5.3 Examples

```
# Extract all files
arcdir_tool EXTRACT data.arc data.dir

# Extract a single file to a custom path
arcdir_tool EXTRACT data.arc data.dir a/b.bin out.bin

# Pack a directory recursively
arcdir_tool PACK data.arc data.dir assets/

# Pack only .bin files from directories
arcdir_tool PACK_BIN data.arc data.dir assets/
```

6 File Discovery and Sorting

When packing files:

1. Analyze each command-line argument;
2. Recursively scan directories;
3. Add individual files directly;
4. Apply optional extension filter;
5. Sort resulting file paths lexicographically.

7 Binary Format Specification

All integer fields are stored in big-endian byte order.

7.1 DIR File Structure

7.1.1 Header

Offset	Size	Description
0x00	4 bytes	Total size of the DIR file (u32, big-endian)
0x04	4 bytes	Number of entries (u32, big-endian)

Table 2: DIR Header Structure

7.1.2 Entry Structure

Each entry:

Field	Size	Description
Offset	4 bytes	Offset in ARC where file data starts (u32, big-endian)
Size	4 bytes	Size of file data in bytes (u32, big-endian)
PathLen	4 bytes	Length of path string including padding (u32, big-endian)
Path	PathLen bytes	Null-terminated path string, padded to 4-byte alignment

Table 3: DIR Entry Structure

7.2 ARC File Structure

- Raw file data written sequentially;
- Padding added after each file to align the next file to 32 bytes;
- Padding bytes filled with `0xCC`.

7.3 Alignment Rules

7.3.1 Path String Padding

```
pad = (~(len - 1)) & (4 - 1);
total_len = len + pad;
```

7.3.2 Data Padding

```
pad = (~(data_size - 1)) & (32 - 1);
```

8 Packing Algorithm

1. Open ARC and DIR files in write mode;
2. Reserve 8 bytes in DIR for the header;
3. For each sorted file:
   - Read file contents into memory;
   - Write DIR entry: ARC offset, file size, path length with padding, path string with padding;
   - Write file data to ARC sequentially;
   - Apply padding to 32-byte alignment using 0xCC.
4. Write total DIR size and entry count at the beginning of the DIR file.

9 Extraction Algorithm

1. Open ARC and DIR files in read mode;
2. Read the DIR header (total size and entry count);
3. For each entry:
   - Read offset, size, and path length;
   - Read path string and remove padding;
   - Apply optional path redirection rules;
   - Seek to ARC offset;
   - Read file data and write to the output path.

10 Path Handling

- Paths are represented as byte strings, compatible with Shift-JIS encoding;
- Directory separators are normalized to '/' when packing;
- Redirection tables are optional and applied during extraction.

11 Error Handling

- Fixed-size reads; EOF triggers an error;
- DIR header size is used for sanity checks;
- I/O errors cause immediate termination.

12 Determinism

For identical input data:

- DIR entries and ARC layout are identical;
- Lexicographic sorting ensures reproducibility;
- Fixed padding rules preserve correct data alignment.

13 Conclusion

The ARC/DIR format is an indexed data container:

- `.arc` stores sequential blocks of raw data;
- `.dir` contains metadata mapping file paths to offsets and sizes;
- Big-endian integers and strict alignment rules are used;
- The `arcdir_tool` utility provides complete packing and extraction with deterministic results.