NAME

gitformat-commit-graph - Git commit-graph format

SYNOPSIS

$GIT_DIR/objects/info/commit-graph
$GIT_DIR/objects/info/commit-graphs/*

DESCRIPTION

The Git commit-graph stores a list of commit OIDs and some associated metadata, including:

- generation number of the commit.
- root tree OID.
- commit date.
- parents of the commit, stored using positional references within the graph file.
- Bloom filter of the commit carrying the paths that were changed between the commit and its first parent, if requested.

These positional references are stored as unsigned 32-bit integers corresponding to the array position within the list of commit OIDs. Due to some special constants we use to track parents, we can store at most (1 << 30) + (1 << 29) + (1 << 28) - 1 (around 1.8 billion) commits.

COMMIT-GRAPH FILES HAVE THE FOLLOWING FORMAT:

In order to allow extensions that add extra data to the graph, we organize the body into "chunks" and provide a binary lookup table at the beginning of the body. The header includes certain values, such as number of chunks and hash type.

All multi-byte numbers are in network byte order.

HEADER:

4-byte signature:

The signature is: {'C', 'G', 'P', 'H'}

1-byte version number:
Currently, the only valid version is 1.

1-byte Hash Version
We infer the hash length (H) from this value:
1 => SHA-1
2 => SHA-256
If the hash type does not match the repository's hash algorithm, the commit-graph file should be ignored with a warning presented to the user.

1-byte number (C) of "chunks"

1-byte number (B) of base commit-graphs
We infer the length (H*B) of the Base Graphs chunk from this value.

CHUNK LOOKUP:

(C + 1) * 12 bytes listing the table of contents for the chunks:
First 4 bytes describe the chunk id. Value 0 is a terminating label. Other 8 bytes provide the byte-offset in current file for chunk to start. (Chunks are ordered contiguously in the file, so you can infer the length using the next chunk position if necessary.) Each chunk ID appears at most once.

The CHUNK LOOKUP matches the table of contents from the chunk-based file format, see linkgit:gitformat-chunk[5]

The remaining data in the body is described one chunk at a time, and these chunks may be given in any order. Chunks are required unless otherwise specified.

CHUNK DATA:

OID Fanout (ID: {O, I, D, F}) (256 * 4 bytes)

The ith entry, F[i], stores the number of OIDs with first byte at most i. Thus F[255] stores the total number of commits (N).

OID Lookup (ID: {O, I, D, L}) (N * H bytes)

The OIDs for all commits in the graph, sorted in ascending order.

Commit Data (ID: {C, D, A, T }) (N * (H + 16) bytes)

- first H bytes are for the OID of the root tree.

- next 8 bytes are for the positions of the first two parents of the ith commit. Stores value 0x70000000 if no parent in that position. If there are more than two parents, the second value has its most-significant bit on and the other bits store an array position into the Extra Edge List chunk.

- next 8 bytes store the topological level (generation number v1) of the commit and the commit time in seconds since EPOCH. The generation number uses the higher 30 bits of the first 4 bytes, while the commit time uses the 32 bits of the second 4 bytes, along with the lowest 2 bits of the lowest byte, storing the 33rd and 34th bit of the commit time.

Generation Data (ID: {G, D, A, 2 }) (N * 4 bytes) [Optional]

- list of 4-byte values store corrected commit date offsets for the commits, arranged in the same order as commit data chunk.

- the corrected commit date offset cannot be stored within 31 bits, the value has its most-significant bit on and the other bits store the position of corrected commit date into the Generation Data Overflow chunk.

- Data chunk is present only when commit-graph file is written by compatible versions of Git and in case of split commit-graph chains, the topmost layer also has Generation Data chunk.

Generation Data Overflow (ID: {G, D, O, 2 }) [Optional]

- list of 8-byte values stores the corrected commit date offsets for commits with corrected commit date offsets that cannot be stored within 31 bits.

- Data Overflow chunk is present only when Generation Data chunk is present and atleast one corrected commit date offset cannot be stored within 31 bits.

Extra Edge List (ID: {E, D, G, E}) [Optional]

> This list of 4-byte values store the second through nth parents for all octopus merges. The second parent value in the commit data stores an array position within this list along with the most-significant bit on. Starting at that array position, iterate through this list of commit positions for the parents until reaching a value with the most-significant bit on. The other bits correspond to the position of the last parent.

Bloom Filter Index (ID: {B, I, D, X}) (N * 4 bytes) [Optional]

- ith entry, BIDX[i], stores the number of bytes in all Bloom filters from commit 0 to commit i (inclusive) in lexicographic order. The Bloom filter for the i-th commit spans from BIDX[i-1] to BIDX[i] (plus header length), where BIDX[-1] is 0.

- BIDX chunk is ignored if the BDAT chunk is not present.

Bloom Filter Data (ID: {B, D, A, T}) [Optional]

- starts with header consisting of three unsigned 32-bit integers:

  - of the hash algorithm being used. We currently only support value 1 which corresponds to the 32-bit version of the murmur3 hash implemented exactly as described in **https://en.wikipedia.org/wiki/MurmurHash#Algorithm** and the double hashing technique using seed values 0x293ae76f and 0x7e646e2 as described in **https://doi.org/10.1007/978-3-540-30494-4_26** "Bloom Filters in Probabilistic Verification"

  - number of times a path is hashed and hence the number of bit positions that cumulatively determine whether a file is present in the commit.

  -

minimum number of bits *b* per entry in the Bloom filter. If the filter contains *n* entries, then the filter size is the minimum number of 64-bit words that contain n*b bits.

- rest of the chunk is the concatenation of all the computed Bloom filters for the commits in lexicographic order.

- Commits with no changes or more than 512 changes have Bloom filters of length one, with either all bits set to zero or one respectively.

- BDAT chunk is present if and only if BIDX is present.

Base Graphs List (ID: {B, A, S, E}) [Optional]

This list of H-byte hashes describe a set of B commit-graph files that form a commit-graph chain. The graph position for the ith commit in this file's OID Lookup chunk is equal to i plus the number of commits in all base graphs. If B is non-zero, this chunk must exist.

TRAILER:

H-byte HASH-checksum of all of the above.

HISTORICAL NOTES:

The Generation Data (GDA2) and Generation Data Overflow (GDO2) chunks have the number *2* in their chunk IDs because a previous version of Git wrote possibly erroneous data in these chunks with the IDs "GDAT" and "GDOV". By changing the IDs, newer versions of Git will silently ignore those older chunks and write the new information without trusting the incorrect data.

GIT

Part of the **git**(1) suite