Heap dump format for other versions:

- [[heapdump13]]
- [[heapdump14]]

Introduction

Go 1.5 has a runtime/debug.WriteHeapDump function that writes all objects in the heap plus additional info (roots, goroutines, finalizers, etc.) to a file. The format of this file is specified here.

Details

The file starts with the bytes of the string "go1.5 heap dump\n". This description also applies to files starting with "go1.6 heap dump\n" and "go1.7 heap dump\n". The go1.6 format is identical to 1.5, and the go1.7 format has one small change described below.

The rest of the file is a sequence of records. Records can be of several different kinds. Records will contain the following primitives:

- uvarint - a 64-bit unsigned integer encoded as in encoding/binary.{Put,Read}Uvarint
- string - a uvarint-encoded length followed by that many bytes of data
- bool - a uvarint-encoded 0 for false or 1 for true
- fieldlist - a description of the pointer-bearing portions of a memory region. It consists of repeated pairs of uvarints encoding a field kind and a field offset, followed by an end-of-list marker. The only possible kind is 1=Ptr. Earlier versions of the heap dump could contain 2=Iface and 3=Eface, but the runtime no longer tracks that information, so it is not present in the dump. Interface values appear as a pair of pointers. 0=Eol is the end of list marker. The end of list marker does not have a corresponding offset.

Each record starts with a uvarint-encoded integer describing the type of the record:

- 0 = EOF
- 1 = object
- 2 = otherroot
- 3 = type
- 4 = goroutine
- 5 = stack frame
- 6 = dump params
- 7 = registered finalizer
- 8 = itab
- 9 = OS thread
- 10 = mem stats
- 11 = queued finalizer
- 12 = data segment
- 13 = bss segment
- 14 = defer record
- 15 = panic record
- 16 = alloc/free profile record
- 17 = alloc stack trace sample

The remaining fields of each record are type-dependent and are described below.

EOF

An EOF record has no fields and must appear last.

object

- uvarint: address of object
- string: contents of object
- fieldlist: describes pointer-containing fields of the object

The size of the contents string is the size of the containing sizeclass, not the size of the object itself. As such, contents size may be somewhat bigger than the contained object's type.

otherroot

- string: textual description of where this root came from
- uvarint: root pointer

type

- uvarint: address of type descriptor
- uvarint: size of an object of this type
- string: name of type
- bool: whether the data field of an interface containing a value of this type has type T (false) or *T (true)

goroutine (G)

- uvarint: address of descriptor
- uvarint: pointer to the top of stack (the currently running frame, a.k.a. depth 0)
- uvarint: go routine ID
- uvarint: the location of the go statement that created this goroutine
- uvarint: status
- bool: is a Go routine started by the system
- bool: is a background Go routine
- uvarint: approximate time the go routine last started waiting (nanoseconds since the Epoch)
- string: textual reason why it is waiting
- uvarint: context pointer of currently running frame
- uvarint: address of os thread descriptor (M)
- uvarint: top defer record
- uvarint: top panic record

Possible statuses:

- 0 = idle
- 1 = runnable
- 3 = syscall
- 4 = waiting

The wait fields must be present in all cases, but they only mean something if the status is "waiting".

stack frame

- uvarint: stack pointer (lowest address in frame)
- uvarint: depth in stack (0 = top of stack)
- uvarint: stack pointer of child frame (or 0 if none)
- string: contents of stack frame
- uvarint: entry pc for function
- uvarint: current pc for function
- uvarint: continuation pc for function (where function may resume, if anywhere)
- string: function name
- fieldlist: list of kind and offset of pointer-containing fields in this frame

dump params

- bool: big endian
- uvarint: pointer size in bytes
- uvarint: starting address of heap
- uvarint: ending address of heap
- string: architecture name
- string: GOEXPERIMENT environment variable value
- uvarint: runtime.ncpu

finalizer

- uvarint: address of object that has a finalizer
- uvarint: pointer to FuncVal describing the finalizer
- uvarint: PC of finalizer entry point
- uvarint: type of finalizer argument
- uvarint: type of object

This finalizer has been registered with the runtime system, but the object to which it refers was either reachable as of the most recent GC or allocated since the most recent GC.

itab

- uvarint: Itab address
- uvarint: address of type descriptor for contained type
    - Up to go1.6, the type is always a pointer type, and represents the type of the itab.data field.
    - From go1.7 and beyond, the type is the type stored in the interface. To decide whether the itab.data field is T or *T requires looking at the last boolean in the referenced type's descriptor.

osthread (M)

- uvarint: address of this os thread descriptor
- uvarint: Go internal id of thread
- uvarint: os's id for thread

memstats

Records the following fields of runtime.MemStats:

- uvarint: Alloc
- uvarint: TotalAlloc
- uvarint: Sys
- uvarint: Lookups
- uvarint: Mallocs
- uvarint: Frees
- uvarint: HeapAlloc
- uvarint: HeapSys
- uvarint: HeapIdle
- uvarint: HeapInuse
- uvarint: HeapReleased
- uvarint: HeapObjects
- uvarint: StackInuse
- uvarint: StackSys
- uvarint: MSpanInuse
- uvarint: MSpanSys
- uvarint: MCacheInuse
- uvarint: MCacheSys
- uvarint: BuckHashSys
- uvarint: GCSys
- uvarint: OtherSys
- uvarint: NextGC
- uvarint: LastGC
- uvarint: PauseTotalNs
- 256 uvarints: PauseNs
- uvarint: NumGC

queuedfinalizer

- uvarint: address of object that has a finalizer
- uvarint: pointer to FuncVal describing the finalizer
- uvarint: PC of finalizer entry point
- uvarint: type of finalizer argument
- uvarint: type of object

This finalizer is ready to run - the object to which it refers is unreachable. The runtime system just hasn't gotten around to running it yet.

data

- uvarint: address of the start of the data segment
- string: contents of the data segment
- fieldlist: kind and offset of pointer-containing fields in the data segment.

bss

Same format as data, but for the bss segment.

defer

- uvarint: defer record address
- uvarint: containing goroutine
- uvarint: argp
- uvarint: pc
- uvarint: FuncVal of defer
- uvarint: PC of defer entry point
- uvarint: link to next defer record

panic

- uvarint: panic record address
- uvarint: containing goroutine
- uvarint: type ptr of panic arg eface
- uvarint: data field of panic arg eface
- uvarint: ptr to defer record that's currently running
- uvarint: link to next panic record

alloc/free profile record

- uvarint: record identifier
- uvarint: size of allocated object
- uvarint: number of stack frames. For each frame:
  - string: function name
  - string: file name
  - uvarint: line number
- uvarint: number of allocations
- uvarint: number of frees

alloc sample record

- uvarint: address of object
- uvarint: alloc/free profile record identifier