Dynamic memory organization

- Considering one possible approach (gnu) to organize dynamic memory on the system side
- Give each application its own unique (virtual) memory space
- Allow that space to be subdivided into space for threads/child processes the application may spawn
- Maintain data with each chunk of memory, free or in use, to track its size and status (support the allocate/free cycle)
- Maintain data structures to organize the chunks of free memory to effectively respond to allocate/free requests

Arenas

- Each application given an “arena”, which controls the memory space for that application
- Arena has it’s associated memory stored in chunks in a heap (it controls division into chunks)
- Arena can subdivide its space into smaller arenas (e.g. use a chunk of its heap to create an arena for a child process) and/or smaller heaps (i.e. use a chunk of its heap to create another, smaller heap, for another purpose)

Heaps

- Heap is section of memory, divided into chunks, each of which can be either free or allocated to application
- Heap can divide chunks into multiple smaller chunks, or coalesce adjacent free chunks into a single larger free chunk as needed
- Big “top” chunk of memory on heap kept free as long as possible, for cases when allocate requests can’t be satisfied from the other chunks

Chunks

- Allocated chunks need to have the space the user requested, plus administrative data to support allocation and free operations
- Free chunks mostly empty, but part of space needs to be used for similar administrative information
- Admin info usually at beginning/end of chunks, so they are easily accessible by allocate/release routines

Allocated chunk admin data

- At start of chunk:
  - Size (bytes)
  - Flag indicating which arena controls it
  - Flag indicating if directly memory mapped (biiiig chunks)
  - Flag indicating if preceding chunk is free or not
- Followed by user content

Free chunk admin data

- At start of chunk:
  - Total size of chunk
  - Arena flag
  - Is it memory mapped
  - Is preceding chunk free or not
  - Pointer to next chunk
  - Pointer to previous chunk
- Most of chunk is unused while free
- At end of chunk:
  - Size (again)