Applicant in the broadest sense, superseding any other implied definitions hereinbefore or hereinafter unless expressly asserted by the Applicant to the contrary, to mean one or more elements selected from the group comprising A, B, . . . and N. In other words, the phrases mean any combination of one or more of the elements A, B, . . . or N including any one element alone or the one element in combination with one or more of the other elements which may also include, in combination, additional elements not listed.

While various embodiments have been described, it will be apparent to those of ordinary skill in the art that many more embodiments and implementations are possible. Accordingly, the embodiments described herein are examples, not the only possible embodiments and implementations.

What is claimed is:

1. A system comprising:

a memory comprising a mapping of a first portion of a memory-mapped file to a virtual address for a first process, wherein the memory-mapped file comprises virtual memory backed by a file; and

a processor configured to:

map a second portion of the memory-mapped file to the virtual address for a second process in response to a forking of the second process from the first process, wherein the first and second portions of the memory-mapped file are backed by the file; and

write data from the first and second portions of the memory-mapped file to corresponding first and second portions of the file that backs the memory-mapped file.

2. The system of claim **1**, wherein the processor is configured to create a copy of data associated with the first portion of the memory-mapped file in response to a write to the virtual address by the first process or the second process.

3. The system of claim **1**, wherein the processor is configured to create a copy of data associated with the first portion of the memory-mapped file when the second process is forked from the first process.

4. The system of claim **1**, wherein the processor is configured to create a copy of data associated with the first portion of the memory-mapped file in response to a memory access operation on the virtual address by the first process or the second process.

5. The system of claim **1**, wherein the file is a pseudo file.

6. A method for providing fork-safe memory allocation from memory-mapped files, the method comprising:

allocating a first portion of a memory-mapped file to a virtual address for a first process;

allocating a second portion of the memory-mapped file to the virtual address for a second process in response to forking the second process from the first process, wherein the first portion of the memory-mapped file and the second portion of the memory-mapped file are mapped to a first offset and a second offset, respectively, of a file that backs the memory-mapped file; and

writing contents of the first and second portions of the memory-mapped file to corresponding first and second portions of the file that backs the memory-mapped file.

7. The method of claim **6**, further comprising selecting the second offset of the file for the second portion of the memory-mapped file when the virtual address is accessed and/or written by the second process forked from the first process.

8. The method of claim **6**, wherein the file includes a first file and a second file, and wherein the first portion of the memory-mapped file is mapped to the first offset of the first file and the second portion is mapped to the second offset of the second file.

9. The method of claim **6**, wherein allocating the first portion of a memory-mapped file comprises allocating the first portion in response to an invocation of an mmap( ) programmatic procedure configured to allocate anonymous memory.

10. The method of claim **6**, wherein allocating the first portion of a memory-mapped file comprises allocating the first portion in response to an invocation of a malloc( ) programmatic procedure configured to allocate anonymous memory.

11. The method of claim **6**, wherein allocating the first portion of a memory-mapped file comprises allocating the first portion in response to an invocation of a brk( ) programmatic procedure configured to allocate anonymous memory.

12. The method of claim **6**, wherein allocating the first portion of a memory-mapped file comprises allocating the first portion in response to an invocation of a sbrk( ) programmatic procedure configured to allocate anonymous memory.

13. An apparatus comprising:

a processor; and

a memory, the memory comprising:

a first virtual address space for a first process executed by the processor;

a second virtual address space for a second process executed by the processor, wherein a virtual address is in both the first virtual address space and the second virtual address space;

at least a subset of a memory-mapped file, the memory-mapped file comprising a first portion and a second portion that include virtual memory backed by a file; and

a memory allocation module configured to map the virtual address in the first virtual address space for the first process to the first portion of the memory-mapped file and to map the virtual address in the second virtual address space for the second process to the second portion of the memory-mapped file in response to a forking of the second process from the first process, wherein the memory allocation module is further configured to write contents of the first and second portions of the memory-mapped file to corresponding first and second portions of the file that backs the memory-mapped file.

14. The apparatus of claim **13**, wherein the memory allocation module is further configured to encrypt data associated with the first portion or the second portion of the memory-mapped file before or as the data is written to the memory-mapped file, wherein the memory-mapped file includes virtual memory assigned to a pseudo file.

15. The apparatus of claim **14** wherein the memory allocation module is further configured to read the encrypted data from the memory-mapped file and decrypt the encrypted data after or as the encrypted data is read from the memory-mapped file.

16. The apparatus of claim **13**, wherein the memory allocation module comprises executable instructions included in an operating system.

17. The apparatus of claim **13**, wherein the memory allocation module comprises executable instructions included in a kernel or kernel module.

18. The apparatus of claim **13**, wherein the memory-mapped file is mapped to an interface configured to provide access over a network to a corresponding area of primary memory of a memory appliance.

19. The apparatus of claim **18**, wherein access over the network to the corresponding area of primary memory of the memory appliance is via client-side memory access in which a communication interface of the memory appliance is configured to access the corresponding area of primary memory of the memory appliance.

20. The apparatus of claim **13**, wherein the first process and/or the second process are included in a container, a jail, and/or a zone.

* * * * *