Tip 78: Use mmap for gigantic data sets

Ben Klemens

6 March 2012

level: heavy lifter
purpose: Analyze 10GB of data with 8GB of memory

I've mentioned the three types of memory (Entry #070) (static, manual, and automatic) before, and guess what—there's a fourth: disk based. Here, we take a file on the hard drive and map it to a location in memory.

This is often how shared libraries work: the system finds `libwhatever.so`, assigns the segment of the file representing a needed function a memory address, and there you go, you've loaded a function into memory.

Or, we could share data across processes by having them both mmap the same file.

Or, we could use this to save data structures to memory. Mmap a file to memory, use `memcpy` to copy your in-memory data structure to the mapped memory, and you're done. Problems come up when your data structure has a pointer to another data structure; converting a series of pointed-to data structures to something saveable is the *serialization* problem, which I'm probably not going to cover (there are libraries that'll help you with it).

And, of course, there's dealing with large data sets. The size of an mmaped array is constrained by the size of your disk, not memory.

OK, sample code. The `load_mmap` routine does most of the work. If used as a malloc, then it needs to create the file and stretch it to the right size; if you are opening an already-existing file, it just has to be opened and mmaped.

```
#include <stdio.h>
#include <unistd.h> //lseek, write, close
#include <stdlib.h> //exit
#include <fcntl.h>  //open
#include <sys/mman.h>

#define Mapmalloc(number, type, filename, fd) load_mmap(filename, &(fd), (number
#define Mapload(number, type, filename, fd) load_mmap(filename, &(fd), (number)*
#define Mapfree(number, type, fd, pointer) releasemmap(pointer, (number)*sizeof(
#define Stopifnot(assertion, ...) if (!(assertion)){printf(__VA_ARGS__); exit(1)

void *load_mmap(char *filename, int *fd, size_t size, char make_room) {
    *fd = open(filename,
```

```
            make_room=='y' ?  O_RDWR | O_CREAT | O_TRUNC : O_RDWR,
            (mode_t)0600);
    Stopifnot(*fd != -1, "Error opening file");

    if (make_room=='y'){ // Stretch the file size to the size of the (mmapped) a
        int result = lseek(*fd, size-1, SEEK_SET);
        Stopifnot(result != -1, "Error calling lseek() to stretch the file");
        result = write(*fd, "", 1);
        Stopifnot (result == 1, "Error writing last byte of the file");
    }

    void *map = mmap(0, size, PROT_READ | PROT_WRITE, MAP_SHARED, *fd, 0);
    Stopifnot (map != MAP_FAILED, "Error mmapping the file");
    return map;
}

void releasemmap(void *map, size_t size, int fd){
    Stopifnot(munmap(map, size) != -1, "Error un-mmapping the file");
    close(fd);
}

int main(int argc, char *argv[]) {
    int fd;
    long int N=1e5+6;
    int *map = Mapmalloc(N, int, "mmapped.bin", fd);

    for (long int i = 0; i <N; ++i) map[i] = i; //you can't tell it's on disk.

    Mapfree(N, int, fd, map);

    //Now reopen and do some counting.
    int *readme = Mapload(N, int, "mmapped.bin", fd);

    long long int oddsum=0;
    for (long int i = 0; i <N; ++i) if (readme[i]%2) oddsum += i;
    printf("The sum of odd numbers up to %li: %lli\n", N, oddsum);

    Mapfree(N, int, fd, readme);
}
```

Final details: the `mmap` function is POSIX-standard, so it is available everywhere but Windows boxes. There, you can do the identical thing but with different function names and flags; see `CreateFileMapping` and `MapViewOfFile`. Glib wraps both `mmap` and the Windows functions in an *if POSIX ... else if Windows ...* construct and named the whole thing `g_mapped_file_new`. You'll have no problem finding

lots of other mmap tutorials online (I referred to this one[1] while writing this).

PS: I've been on the fence about implementing an `apop_data_alloc_via_mmap` sort of function for a while. It really wouldn't be all that hard, but would add one more moving part. Am open to opinions on this.

[1] `http://www.linuxquestions.org/questions/programming-9/mmap-tutorial-c-c-511265/`