Memory Management

1. open - Link 1
    - `int open(const char *pathname, int flags);`
    - The `open()` system call opens the file specified by pathname. The argument `flags` must include one of the following access modes: O_RDONLY, O_WRONLY, or O_RDWR. These request opening the file read-only, write-only, or read/write, respectively.
2. fstat - Link 1, Link 2 - Definition of `struct stat`
    - `int fstat(int fildes, struct stat *buf);`
    - The `fstat()` function shall obtain information about an open file associated with the file descriptor fildes, and shall write it to the area pointed to by `buf`.
3. mmap - Link 1, Link 2
    - `void *mmap(void addr[.length], size_t length, int prot, int flags, int fd, off_t offset);`
    - `mmap()` creates a new mapping in the virtual address space of the calling process. The starting address for the new mapping is specified in `addr`. The `length` argument specifies the length of the mapping (which must be > 0).
    - More about `mmap` will be covered in the tutorial sheet.
4. write - Link 1
    - `ssize_t write(int fd, const void buf[.count], size_t count);`
    - `write()` writes up to `count` bytes from the buffer starting at `buf` to the file referred to by the file descriptor `fd`.
5. munmap - Link 1
    - `int munmap(void *addr, size_t len);`
    - The munmap() function shall remove any mappings for those entire pages containing any part of the address space of the process starting at addr and continuing for len bytes.

Problem 0:

Create a C program that dynamically allocates an array of integers using `mmap` . The program should create an array of size N and initialize the array elements. After initializing the array, print the elements from the memory to the console.

Ans:

```
#include <stdio.h>
#include <sys/mman.h>

int main(){

  int N=5;
  int *ptr = mmap ( NULL, N*sizeof(int),
```

```
      PROT_READ | PROT_WRITE, MAP_PRIVATE | MAP_ANONYMOUS, 0, 0 );

  if(ptr == MAP_FAILED){
    printf("Mapping Failed\n");
    return 1;
  }

  for(int i=0; i<N; i++)
     ptr[i] = i*10;

  printf("The elements of the array => ");
  for(int i=0; i<N; i++)
     printf("[%d] ",ptr[i]);

  printf("\n");
  int err = munmap(ptr, 10*sizeof(int));
  if(err != 0){
     printf("UnMapping Failed\n");
     return 1;
  }

  return 0;
}
```

Recommended reads for more information:

https://www.golinuxcloud.com/tutorial-linux-memory-management-overview/
https://www.baeldung.com/linux/file-system-caching
https://www.linuxfordevices.com/tutorials/linux/memory-management-linux