CS446 Lab 9 – Memory Mapped files

Labs that are not scheduled for a Lab Test are not mandatory. These are practice labs, designed to help you on your lab test and assignment.

Solutions to practice labs will not be posted online.

Outline

In this lab you are going to study Memory Mapping files in Linux.

https://www.ibm.com/docs/en/aix/7.2?topic=memory-understanding-mapping - has a good explanation for memory mapped files.

https://man7.org/linux/man-pages/man2/mmap.2.html - has a good explanation `mmap()` system call.

You are to write a C program that takes a text file named **characters.txt** (at the command line), opens this file, and maps it to a memory region using the system call **`mmap()`.** The file **characters.txt** contains two characters, separated by **two spaces**, per each line of the file. This file can be downloaded from Canvas. After memory mapping this file, you are to read the contents (one character at a time) and copy the contents to a character array using the `memcpy()` function. Finally, you are to print the contents of this array.

In particular your program needs to do the following:

1. To be able to use `mmap()` system call and the `memcpy()` function you need to add the following header files in addition to your standard input output header file.

```
#include <sys/mman.h> /*For mmap() function*/
#include <string.h> /*For memcpy function*/
```

2. Other useful header files are listed below. They enable you to use the **`open()`** system call which is used to open a new file and obtain its file descriptor.

```
#include <fcntl.h> /*For file descriptors*/
#include <stdlib.h> /*For file descriptors*/
```

3. Define global variables for the signed character array and a signed character pointer to store the starting address of the memory mapped file.

   E.g.:
   ```
   signed char buffer[MEMORY_SIZE];
   signed char *mmapfptr;
   ```

   You can define 'MEMORY_SIZE' as a macro definition.

4. Open the file **(characters.txt)** given as an input to the command line argument using the **`open()`** system call. Since you will be simply reading this file use the **O_RDONLY** option.

   E.g.: `int mmapfile_fd = open(argv[1], O_RDONLY);`

5. Use the **`mmap()`** system call to memory map this file.

   E.g.: `mmapfptr = mmap(0, MEMORY_SIZE, PROT_READ,MAP_PRIVATE, mmapfile_fd, 0);`

6. Retrieve the contents of the memory mapped file (using a loop) and store it in the signed character array using **`memcpy()`** function. Sample code to use `memcpy()` is:

   ```
   memcpy(buffer + i, mmapfptr + temp, CHAR_SIZE);
   ```

   `CHAR_SIZE` = the size of the contents in bytes to be copied from the memory mapped file to **buffer**. Since we are reading only one character at a time,
   `CHAR_SIZE` = 1.
   `temp` = offset in bytes from the base address stored in **`mmapfptr`**

7. Unmap the memory mapped file using the **`unmap()`** system call.

   E.g.: `munmap(mmapfptr, MEMORY_SIZE);`

8. Compile your program without errors.

Sample Program output: **./lab9 characters.txt**

Operating Systems!

Note:

Refer to lecture notes of Virtual Memory (Chapter 10), particularly slides # (18 - 21) for this practice lab.