Lebanese American University
School of Arts and Sciences
Department of Computer Science and Mathematics

CSC326 – Operating Systems
Dr. Abdallah Dabboussi

Lab Assignment 6

Problem 1:

In this problem, we are interested in sharing an array of integers between a child and a parent processes. Write a C program that:

1. Allocates a shared array of N integers where the size of the array N will be given as a command line argument. Your program should check if the shared array was created successfully or mmap failed.
2. Scans N integers from the user and fills the array
3. Forks a child that computes and displays the average of all the elements of the array.
4. The child then scans from the user an integer a and then modifies the array by multiplying the elements which are even integers by a
5. The parent process should wait for the child process to terminate and then displays the array
6. The parent then modifies the elements of the array by squaring the odd integers. The parent process should then display the updated elements of the array.

Note: You need to implement a function printArray to display the elements of the array. You may need also to free the shared memory at the end of your program.

Sample Output: (Input = ./test 5)

```
Length N=5
11
4
13
20
5
From child:
Please enter a:
```

```
100
11 400 13 2000 5
From Parent:
11 400 13 2000 5
121 400 169 2000 25
```

- *Submit your solution in a file called "Problem1.c".*

<u>Problem 2:</u>

Write a C program that creates a child process and uses shared memory as a mean of communication between the processes. Your program should:

1. Allocate a shared array of 5 strings with a maximum string size of 20
2. Fork a child that reads 5 names from the user and save them in the array.
3. The parent process should wait for the child process to terminate and then display every string, its length and the number of lower case characters

Note:

- *In order to have the five strings in one array, you may need first to allocate a shared memory for the whole array char* $**shared\_array$ *and then allocate shared memory for each element* $shared\_array[i]$ *in the array. You may need also to free the shared memory at the end of your program.*
- *In case you need to copy a string to a temporary char* $text[20]$*, you may need to use* $strcpy(text, shared\_array[i])$ *to copy the string* $shared\_array[i]$ *to* $text$

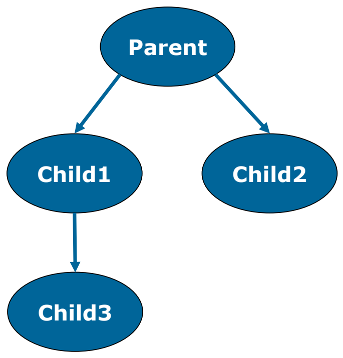
Sample Output:

```
From child:
Please enter a name 1:Omar
Please enter a name 2:LEA
Please enter a name 3:Rami
Please enter a name 4:Alice
Please enter a name 5:BOB

From Parent:

*** String 1 ***
Omar
Length = 4
The number of uppercase letters is: 1
The number of lowercase letters is: 3

*** String 2 ***
LEA
Length = 3
The number of uppercase letters is: 3
The number of lowercase letters is: 0

*** String 3 ***
Rami
Length = 4
The number of uppercase letters is: 1
The number of lowercase letters is: 3

*** String 4 ***
Alice
Length = 5
The number of uppercase letters is: 1
The number of lowercase letters is: 4

*** String 5 ***
BOB
Length = 3
The number of uppercase letters is: 3
The number of lowercase letters is: 0
```

- *Submit your solution in a file called "Problem2.c".*

Problem 3:

Write a C program that will create three processes that share an array of N elements as follows:



- Scan the length of the array of integers N from the user and create a shared array between the different processes
- The process to be executed first (child 3) will scan the integers from the user and fill the array

- The second process to be executed (child 1) will multiply the array elements by 10
- The third process (child 2) will deduct 2 from every element
- The parent will then compute the sum of the elements of the array. The parent should then expand the existing memory mapping for the array to add the sum at the last position of the array.

*Note: You may need to use the mremap() function (void * mremap(void * old_address, size_t old_size, size_t new_size, int flags);) which expands (or shrinks) an existing memory mapping, potentially moving it at the same time, depending on the flags argument and the available virtual address space. You may set the flag to 0. In order to use the mremap() method, you need to add (#define_GNU_SOURCE) before including any headers.*

Sample Output:

```
Please enter the size of the array:
4
*** From Child 3: ***
Please enter 4 integers:
10
11
12
13
10 11 12 13

*** From Child 1: ***
100 110 120 130

*** From Child 2: ***
98 108 118 128

*** From Parent: ***
98 108 118 128 452
```

- *Submit your solution in a file called "Problem3.c".*

<u>Problem X:</u>(Optional)

Write a C program that creates 3 processes (P1, P2, and P3). The 3 processes should increment the value of an integer declared as a shared memory using mmap() as follows:

1. P1 should loop to increment the integer from 0 to 10000
2. P2 should loop to increment the integer from 10000 to15000

3. P3 should loop to increment the integer from 15000 to 20000
4. Each process should print the value of the integer before exiting

Note: You are not required to do any sort of synchronization in this question. You can simply use the easy busy waiting technique. You may need also to consider the order of execution of the different processes by using wait or waitpid methods.
Sample Output:

```
Child 1 10000
Child 2 15000
Parent 20000
```

kali • *Submit your solution in a file called "ProblemX.c".*