CENG305 Operating Systems - Spring 2017 Labwork 3 **Nov 25 – Dec 16 , 2017**

Question 1

Write a C program which takes two arguments from command line: one filename and one number of items. Your program must check if correct number of arguments are supplied and terminate if not. Your program must create three child processes in the following order: first child process creates the mentioned file and fills it with the specified number of random integers (0-1000), one number per line. The second child process executes "sort" command to sort the numbers in this file on screen. The third child process must find the minimum and maximum numbers in this file.

Between each child process, parent must wait for termination of former process to create the new process.

Sample Run

```
$ ./question1 nums.txt 15
[PARENT] Creating first process...
[CHILD1] Creating nums.txt with 15 integers...
[PARENT] Creating second process...
[CHILD2] Executing sort command...
56
133
147
. . .
[PARENT] Executing third process...
[CHILD3] Min: 56, max: 839
[PARENT] Done.
```

Question 2

Write a C program which takes two arguments from command line: one filename and one number of items. Your program must check if correct number of arguments are supplied and terminate if not. Your program must create three child processes in the following order: first child process creates the mentioned file and fills it with the specified number of random characters (a-z). The second child process executes “zip command to make a compressed copy of your file. The third child process must execute “ls” command to display both files.

Between each child process, parent must wait for termination of former process to create the new process.

Sample Run

```
$ ./question1 chars 40000
[PARENT] Creating first process...
[CHILD1] Writing 40000 random chars to chars.txt...
[PARENT] Creating second process...
[CHILD2] Executing zip command...
  adding: chars.txt (deflated 37%)
[PARENT] Creating third process...
[CHILD3] Executing ls command...
40000 chars.txt
25492 chars.zip
[PARENT] Done.
```

Question 3

Write a C program which takes four command line arguments: two file names and two number of elements. Your program must check if correct number of arguments are supplied and terminate if not. Your main process must create three processes: The first and second processes must open / create files by the given names and fill them with the given number of random integers (one integer per line) and terminate. The third process must execute “sort” command with necessary arguments so that it numerically sorts both files and display the combined result on screen.

Before creating the third child process, main process must wait for the other two child processes to end.

Sample Run

```
$ ./question1 randfile1.txt randfile2.txt 40 70
[PARENT] Creating first process...
[PARENT] Creating second process...
[CHILD1] Writing 40 random integers to randfile1.txt... (78 \n 32 \n 56...)
[CHILD2] Writing 70 random integers to randfile2.txt... (46 \n 88 \n 45...)
[PARENT] Creating third process...
[CHILD3] Sorting both files:
32 \n 45 \n 46 \n 56 \n 78 \n 88 ...
[PARENT] Done.
```

Question 4

Write a program in C which creates 10 POSIX threads. These threads must individually generate arrays of 1.000 random integers between 1.000 and 9.999 (inclusive) and count how many of these numbers are prime numbers.

Sample Run

```
$ ./program1
Thread 1 - 1 primes.
Thread 0 - 1 primes.
Thread 3 - 1 primes.
Thread 5 - 2 primes.
Thread 9 - 1 primes.
Thread 2 - 0 primes.
Thread 6 - 1 primes.
Thread 4 - 1 primes.
Thread 8 - 3 primes.
Thread 7 - 0 primes.
```

Question 5

Now alter the program in Question 4, such that now the main thread creates a single array of 10.000 random integers between 1.000 and 9.999 and each thread gets one tenth of this array (instead of generating their individual arrays).

Question 6

Write a program in C which creates 5 POSIX threads. These threads must individually generate arrays of 2.000 random integers between 0 and 50.000 (inclusive) and classify these integers according to their number of digits.

Sample Run

```
$ ./program1
Thread 0: 1-9: 0, 10-99: 7, 100-999: 34, 1000-9999: 358, 10000-99999: 1601
Thread 2: 1-9: 0, 10-99: 5, 100-999: 38, 1000-9999: 366, 10000-99999: 1591
Thread 3: 1-9: 0, 10-99: 2, 100-999: 35, 1000-9999: 373, 10000-99999: 1590
Thread 1: 1-9: 0, 10-99: 2, 100-999: 24, 1000-9999: 356, 10000-99999: 1618
Thread 5: 1-9: 1, 10-99: 5, 100-999: 45, 1000-9999: 346, 10000-99999: 1603
```

Question 7

Now alter the program in Question 6, such that now the main thread creates and fills a single array of 10.000 random integers and each thread gets one fifth of this array (instead of generating their individual arrays).

Question 8

Write a program in C which creates 10 POSIX threads. These threads must individually generate arrays of 5.000 random lowercase characters (a-z) and count number of vowel and consonant characters.

Sample Run

```
$ ./program1
Thread 140026609727232: Vowels: 364, consonants: 4636
Thread 140026601334528: Vowels: 370, consonants: 4630
Thread 140026592941824: Vowels: 386, consonants: 4614
Thread 140026584549120: Vowels: 406, consonants: 4594
Thread 140026576156416: Vowels: 395, consonants: 4605
Thread 140026567763712: Vowels: 385, consonants: 4615
Thread 140026550978304: Vowels: 403, consonants: 4597
Thread 140026534192896: Vowels: 430, consonants: 4570
Thread 140026542585600: Vowels: 416, consonants: 4584
Thread 140026559371008: Vowels: 385, consonants: 4615
```

Question 9

Now alter the program in Question 8, such that now the main thread creates and fills a single array of 50.000 random characters and each thread gets one tenth of this array (instead of generating their individual arrays).