ICS 53, Winter 2018

Lab 2: Search an Array

You will write a program which searches an array of integers to find a given integer. The program should print out the array index where the given integer is found, or it should print -1 to indicate that the integer was not found in the array. Array indices begin at 0. Your program will accept two arguments at the command line, 1) the name of the ***array file*** which contains the array, and 2) the integer to search for in the array. Assume that there will be no duplicated elements in the array.

Multiple Processes

The unique aspect of this search is that it will use multiple processes to search different parts of the array. The parent process of your program will recursively create child processes to search subsets of the array. Each child process may also create child processes to search smaller subarrays. This splitting will continue until the portion of the array being searched by a process is only a single element. At this point the process will compare the single element to the integer being searched for. Each element in the array should be compared to the given integer, and each individual array element should be compared by a different process.

Here is an example of the processes might be created to search an array 5, 6, 7, 8 for an integer. Suppose that your program initially runs in process P1, then P1 is assigned to search the whole array 5, 6, 7, 8. P1 creates two children, P2 which is assigned to search 5, 6, and P3 which is assigned to search 7, 8. P2 creates two children, P4 which is assigned to search 5, and P5 which is assigned to search 6. P3 creates two children P6 which is assigned to search 7, and P7 which is assigned to search 8. Notice that no more splitting will occur because P4, P5, P6, and P7 are all assigned to search single elements of the array.

Output of your Program

Each child process which is examining an individual element of the array should print its process ID number and the value of the array which it is evaluating in the following format, “pid XXXX, value: YYYY”. The initial parent process should also print the final result of the search in the form “Search output: X” if the integer is found in the array, and it should print “Search output: -1” if the integer is not found in the array. Note that the final result output line should be printed at the end since the final result should not be known until all of the individual array elements have been evaluated.

The following is an example of how your code will work when it is called with an array file called ***test3.txt*** which contains the following array, **5 6 7 4**. In this example, we assume that the linux prompt is the symbol '**$**'. We assume for this example that the executable is named "a.out".

```
$ ./a.out test3.txt 5
pid 7470, value: 5
pid 7469, value: 7
pid 7471, value: 4
pid 7472, value: 6
Search output: 0

$ ./a.out test3.txt 8
pid 7528, value: 7
pid 7530, value: 6
pid 7529, value: 5
pid 7531, value: 4
Search output: -1
```

Array File Format

The array file is a file which contains a series of integers, each of which is separated by space character. The array file should have only one line. Any lines after the file line will be ignored.

In the example above we used ***arr_file.txt*** as the name of the array file, but your code should be able to accept an array file with any name.

Execution Warning

This program can create a large number of processes so you can assume that your program is never executed with an array containing more than 10 elements.

Submission Instructions

There will be a dropbox on Canvas which you will use to submit your code. You should submit a single C source code file. **The code must compile and execute on the openlab machines**. The name of your C source code file should be "Lab2.c".