Istanbul Technical University

November 10, 2021

ISE302 Operating Systems
ASSIGNMENT - 1

Due Date: Thursday, November 30, 2021, 23:59.

• Please write and draw neatly in your report and add comments in your source code.

• **Consequences of plagiarism:** Any cheating will be subject to disciplinary action.

• **No late submissions** will be accepted.

• **Submissions:** Submit your report and source code(hw1.c) to Ninova. Please do not forget to write your full name (first name and last name) and Student ID in your report as well as in your source code.

If you have any questions, please e-mail teaching assistant **Esin Ece Aydın (aydinesi16@itu.edu.tr)**.

1 (30 points) Please investigate the given code below. Compile and run the program, and answer the following questions accordingly.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
	int returnValue= 0;

	int i;
	for (i = 0; i < 3; i++){
		returnValue = fork();
		if(returnValue == -1){
			exit(0);
		}
		else if(returnValue != 0){
			wait(NULL);
		}
		printf("Current process id: %d\n", getpid());
	}

	return 0;
}
```

a) (5 points) How many times will the system call *fork()* be called?

b) (5 points) What will the program's output look like?

c) (10 points) How many processes will the program end up with in total? How many of them can be identified as parent and as child processes?

d) (10 points) Draw a tree that represents the hierarchy of the created processes

2(70 points) In this part of the homework, you are asked to write a program using multiple threads to determine the largest element in an integer array. In addition you need to evaluate the performance of your program in terms of time-complexity.

First of all, the main process will create **N** number of threads. Then, each thread will first be asked to find the maximum number in the array's equal-sized sub-parts. The main process will then determine the largest one by examining the numbers returned by each thread, and print the largest element to the console.

For example, assume that there are 5 threads working and the size of the array is 50. Accordingly, each thread will try to find the largest number out of 10 numbers. The start and end indices of corresponding sub-parts of each thread are as follows:

THREAD 0	THREAD 1	THREAD 2	THREAD 3	THREAD 4
[0] ... [9]	[10] ... [19]	[20] ... [29]	[30] ... [39]	[40] ... [49]

In this example, THREAD 0 will check the numbers from the first element to the tenth element of the array and try to find the largest number, while THREAD 1 will check the numbers from the eleventh element to the twentieth element of the array and try to find the largest number in that interval.
The numbers found by each thread will be passed to the main process through the `pthread_join()` function. Since there are 5 threads in this example, the main process will compare among 5 numbers and print the maximum number to the console.

Please follow the instructions in the comment lines in the given skeleton code and make the necessary changes. To see how thread usage affects the total running time of the program, observe the results by using the `time` command, giving the thread count (**N**) 1, 10, 100, 1000, 100000, and 200000 at each run.

The following commands can be used to compile and run the program:

```
$ gcc assignment1.c -o assignment1 -pthread -w
$ time ./a.out
```

Besides the source code, you are expected to explain the source code you written and the statistics you have obtained by running your source code in the report.

Here, when `time` command is used, 3 statistical information is obtained: the actual time (seconds) elapsed, the number of CPU seconds spent on the instruction in user mode and kernel mode.