Istanbul Technical University

March 26, 2024

BLG312 Compyuter Operating Systems
ASSIGNMENT - 1

Due Date: Sunday, April 7 2024, 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 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.

In your report, answer the questions in Section 1 and for Section 2. For Section 2, describe the functions and system calls you used, clarify the logic you employed for given problem, and share your general observations.

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

1 (35 points) Please investigate the given code below. Compile and run the program, and answer the following questions accordingly.

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    printf("Parent Process ID: %d\n", getpid());

    for (int i = 0; i < 4; ++i) {
        pid_t d = fork();

        if (d == -1) {
            perror("Fork failed");
            exit(EXIT_FAILURE);
        } else if (d == 0) {
            printf("Child %d: Process ID %d (Parent: %d)\n", i + 1, getpid(),
getppid());

            for (int j = 0; j <= i; ++j) {
                pid_t d = fork();

                if (d == -1) {
                    perror("Fork failed");
                    exit(EXIT_FAILURE);
                } else if (d == 0) {
                    printf("Child %d: Process ID %d (Parent: %d)\n", j + 1,
getpid(), getppid());
                    exit(EXIT_SUCCESS);
                }
            }
            exit(EXIT_SUCCESS);
        } else
            wait(NULL);
    }
    return 0;
}
```

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

b) (10 points) How many processes will the program create in total? Identify and differentiate between parent processes and child processes in the context of this program.

c) (10 points) Draw a hierarchical tree diagram representing the relationships between the parent and child processes created by the program. Include process IDs in the tree to illustrate the parent-child associations clearly.

d) (10 points) Discuss the significance of using the exit() system call in a multi-process program. Explain what purpose `exit()` serves and how its absence might affect the program's execution, particularly when creating multiple processes using `fork()`.

2 (65 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.

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.

Besides the source code, you are expected to explain the 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.

The following commands can be used to compile and run the program with pthread support while suppressing warnings:

```
$ gcc assignment.c -pthread -w
```

```
$ time ./a.out
```