Introduction to Operating Systems
METU Computer Engineering

# Programming Assignment # 1

**Instructor: Yusuf Sahillioğlu**

**Deadline: 25.04.2021 23:59**

**(20% of the actual grade)**

---

**Your code will be tested by Moss against cheating attempts, any cases suspected of plagiarism will result in total loss of grade and might result in further disciplinary actions.**

**Please submit your code as a C file on ODTUCLASS before the deadline.**

---

You'll work on process creation and communication in Unix using fork, wait, pipe, kill, signal.

***Part 1 [40 points]:*** Use files to pass information from children processes to the parent process. Your parent process is executed via *part1 n* to use the first *n* input files in the file folder, each named input<i>.txt. Parent process creates *n* children processes and each child process sorts the numbers in the input file and writes the result to an intermediate output file. Input file format:
<m>
number_1 number_2 number_3 … number_m

Output file is a 4-liner with the following format:
<m>
sortednumber_1sortednumber_2 sortednumber_3 … sortednumber_m
<execution time in seconds>
<name of the signal received, e.g., SIGUSR2, and receive time, e.g., 11:37:44>

Child process uses *SelectionSort* if its process id is odd and *InsertionSort* otherwise. It also sleeps *x* seconds after the sorting as it gets tired. *x* is a random number between 1 and 7. Its execution time is sorting + sleeping, so it will be at least 1 second.

Parent reads the intermediate files when all children processes finish (*hint*: wait() in Slide 42). It also sends a signal to each active child. Send SIGUSR1 if child id odd, SIGUSR2 if it's even.

Once all intermediate files are read, parent process creates a single output file called output.txt which will be sorted w.r.t. the <execution time>s. Here is its format:
<execution t of proc_i> <sorted numbers for proc_i> <name & t of receive for signal by proc_i>
<execution t of proc_j> <sorted numbers for proc_j> <name & t of receive for signal by proc_j>
and so on.

***Part 2 [40 points]:*** Implement the same program using pipes instead of files.
***Part 3 [20 points]:*** Signal support in both parts, i.e., do the <signal by proc>.