Laboratory 2 (Deadline 15th Jan 2023, 11.59PM)

Individual assignment

Part I

Print the string “Hello World” on screen. Each character must be printed by a different process. The process that prints the i^{th}letter must have been spawned by the process that printed the $(i-1)^{th}$letter. Each process must first print its designated character, as well as its own process ID, second, sleep for a random number of seconds (from 1 to 4 seconds), and then, do anything else it must do to achieve the given task.

What is the minimum lines of C code with which you can achieve the above?

Remember these processes have to be run in the guest Minix3 system. To compile in the guest system, we use a C compiler named *clang* (instead of *gcc*). Also, standard *make* is invoked as *gmake* in the Minix3 environment.

Submit: a single zip file (format: <roll-number>_lab2_part1.zip) with all required source files and a Makefile. The evaluator will simply unzip the submission, and run “gmake hello”. If the desired output is not seen, you will not be awarded any marks.

Part II

Write a collection of programs *twice, half, square* such that they execute sequentially with the same process-id, and each program should also print its PID. *(process id)* The user should be able to invoke any combination of these programs, to achieve the required functionality.

For example consider three programs *twice, half, square* which accept only one integer as argument and does some specific operation.
$twice 10 prints **20** and some int which is its process-id as output
$square 10 prints **100** and some int which is its process-id as output
$half 10 prints **5** and some int which is its process-id as output

Now the user should be able to combine these programs in any combination to achieve the required result.

For example:

$twice square half twice half 10
should calculate *half(twice(half(square(twice(10)))))* and print **200** as result. It should also print the process ids of each program as it executes. **Note that the process-id printed by each of these programs should be the same, in this case.**
$square twice 2
should calculate *twice(square(2))* and print **8** as result, and the process id of square and twice, which should be the same.
The evaluation order is from left to right

Note that the last argument is the integer, and the remaining arguments are the programs to be invoked.

This should be generally applicable to any *n* number of processes, all of which are written by you. *minimum of three should be written*

pl p2 p3 ... pn arg_value

Sample output:

$ ./twice ./square ./half ./twice ./half 10
Twice: Current process id: 9668, Current result: 20
Square: Current process id: 9668, Current result: 400
Half: Current process id: 9668, Current result: 200
Twice: Current process id: 9668, Current result: 400
Half: Current process id: 9668, Current result: 200

Hint: Use execvp family of system calls.

Submit: a single zip file (format: <roll-number>_lab2_part2.zip) with all required source files and a Makefile. The evaluator will simply unzip the submission, and run. If the desired output is not seen, you will not be awarded any marks.

Part III
Modify the Minix3 source code such that:

- A message “Minix: PID <pid> created” is printed, whenever a process is created. (Let us follow the convention throughout this course that anything printed by the Operating

System code will be prepended by the string “Minix: “.)

- A message “Minix: PID <pid> exited” is printed, whenever a process ends.

Comment on the order in which processes are created and processes exit and justify whether is it as expected.

Submit: a single zip file (format: <roll-number>_lab2_part3.zip) with all modified source files and a shell script. The shell script must copy the modified source files to the correct directories, and build the system. The evaluator will simply run the shell script, reboot the system, and check for the desired behavior.

Hint: look at *minix/servers/pm* .

Important: Remember to create a branch in the git repository for this assignment.