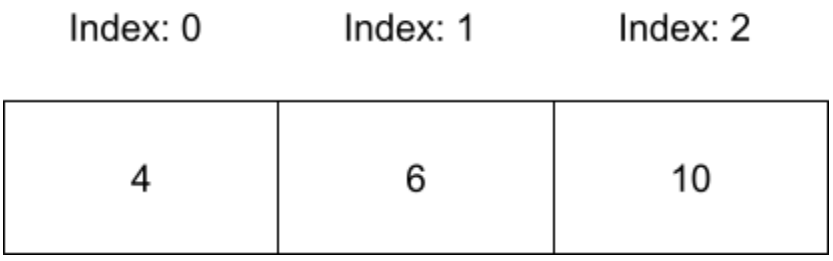
Online 2 Practice Problems

YOU'RE HIRED(or maybe NOT) !

The Placement Unit has got all the students fired up after several training sessions along the weekends. AmongOS, a systems company has come for a campus hiring drive and it's time to put the students' abilities to test. The recruiters need your help to differentiate the true interns from the impostors.

The AmongOS process (Main process) creates a shared memory, followed by the creation of a recruiter process. The recruiter then updates the shared memory with the question. In this scenario, the question is to find the sum of two numbers. The numbers will be present in the first 2 locations of the shared memory (The recruiter will randomly generate 2 numbers and update the shared memory with those numbers). The recruiter then creates a student process to answer the question. The student finds the required sum and then multiplies the sum with a randomly generated 0 or 1. The student then has to update the third location of the shared memory with the answer(after multiplying by 0 or 1). Design a multi-process program in C language that implements this mechanism. Your program should demonstrate the ability to establish shared memory and facilitate inter-process communication between the recruiter and student processes.



A. Implement the above situation WITHOUT using signals. The recruiter should wait until the student exits in order to access the answer generated by the student. The recruiter prints all the values in the shared memory(question and answer). If the answer was correct, the recruiter prints “<PID> was not an impostor ”, otherwise “<PID> was an impostor”. Here PID is the process ID of the student.

B. In continuation to the above question, consider a similar process tree for this question. In this case the recruiter creates the student process as soon as the recruiter process is created. Note that the student process should be in a blocked state until the question has been set by the recruiter. After this the recruiter sets the question (that is generates 2 random numbers) and updates it in the shared memory, sends an appropriate signal to the student to resume execution and waits until the student responds back with the answer. The student calculates the answer(and then multiplies it with 0 or 1) and responds back to the recruiter through the exit() statement. The recruiter collects the value and prints the result as in part A.

C. Consider the above process tree and the above scenario once again! Only thing that changes is the recruiter question format. Instead of finding the sum, the student has to find the factorial of a number. However, this time around you are not allowed to use shared memory and implement everything mentioned above using SIGNALS only. The recruiter sends a single number(in the range 0-10 using sigqueue) and the student calculates(and then multiplies it with 0 or 1) and returns the factorial of the number(again using sigqueue). Print the output in the same format as in part A.

Follow-up:- Generalize the above implementation for N-students case where N is taken as CLI argument and instead the AmongOS process prints the PID of selected students(Hint: Use shared memory again!)

Note: To generate random numbers in a range, refer to the tutorial code on shared memory.