Week 5 - Shared Memory

- 1. Finish all previous work.
- 2. Compile, build and run the code from slide 9 and 10 (program 1 and 2).
  - Note that the memory is not deleted between executions if you run program 2 multiple times. Add the line `shmctl (shmid, IPC_RMID, 0)` to program 1 (before the exit) and run the programs, what happens?
- Now Remove that line and add it to program 2, what happens now (hint: you may need to execute program 2 multiple times to see the change).
  - What does this function do?

Week 5 - Shared Memory

- 3. Write a program that creates a shared memory portion to fit 20 integers. Have one program fill the integers with multiples of `5` (e.g. `0`, `5`, `10`, `15` etc) and another that reads from it (the programs can be executed one after the other).

Week 5 - Shared Memory

- 4. Write a program that creates two portions of shared memory, one to store a single integer and one to store `256` chars.
- The first program should get text from a user using cin to populate a string, it should then place the length of the string in the shared memory integer and copy the contents of the string into the shared memory (use a loop over the string length).
- The second program should have an infinite loop that checks the shared memory integer, if it is non-zero it should read the specified number of bytes from the shared memory character array and output them to the screen. It should set counter to zero once it's read from it.

Week 5 - Shared Memory

- The two programs should be executed at the same time.
  - Did you encounter any problems implementing the above programs?

Week 5 - Shared Memory

- 5. Ensure you understand the code in the consumer-producer example.
- 6. Ensure you understand the code on blackboard for the client-server problem using named pipes.