## ***sfork***: A fork with flexible shared memory regions

### **Team:** RPC (Chinmay, Prashant, Rishik)

**Aim:** In this project we plan to implement a variant of the fork where some part of the memory of the parent can be shared with the child process in a manner such that the update to that memory region from the parent is visible to the child and vice-a-versa. This sharing can be either **unidirectional**, i.e., only the parent can update, and the child can read, and vice-versa, or **bidirectional**, i.e., both the parent and child can update and see the updated values. As we know, it can also be achieved with the help of the shared memory construct. Hence, we will compare sfork with shared memory to study the effectiveness.

**Motivation**:

The shared memory segment approach that is enabled by the header file *`#include <sys/shm.h>`* provides ways for interprocess communication (IPC) between various processes. The processes can read to the shared memory and write to this shared memory region. The access limitation is set by the flags that are provided during creation of the shared memory region. **sfork** provides a method for interprocess communication between a parent process and a child process. The added benefit of using sfork is that for the user the semantics of sfork would be pretty similar to the normal fork call, with the addition of the sharing direction flag i.e. whether the shared memory region is write accessible for parent, child or for both. We aim to do a comparative study to study the pros and cons of a system call as sfork in comparison to the already existing methods of shared memory regions.

**Questions to be answered**:

* The added overheads of creating a shared memory region when the region is shared only between a child process and a parent process
* The benefits of using sfork in terms of the ease of implementation and POSIX compliance of such a system call
* The added complexity needed to set the boundaries of write access to the shared memory region between the parent and the child and how to handle these directional access flags.

**Proposed Milestones**

* **Milestone1 (21st March):**
  + An in depth study into the implementation of the shared memory regions and any possible drawbacks of this approach
* **Milestone2 (30th March):**
  + Beginning the implementation of the syscall sfork in accordance with POSIX rules
* **Milestone3 (12th April):**
  + Complete the implementation of sfork along with the testing.
  + Creating in depth test cases to check for the correctness of the sfork program and making use of any Linux test-suite if any relevant test case can be modified to the required needs
* **Milestone4** **(26th April):**
  + Comparing the performance of sfork with shared memory region approach and analysis the bottleneck of both the approaches
  + Analysing the test cases favourable to each of the two approaches compared
  + Completing the report and the final project submission