Emre Tileylioğlu
202071219

SHARED MEMORY:

POSIX Inter-process communication (IPC) is widely applied through two methods, namely message queue and shared memory. IPC can also be achieved by using special files such as pipes and named pipes. This report briefly explains the advantages and disadvantages of shared memory with respect to other IPC methods in terms of the usability/flexibility, efficiency, reliability and security topics.

The pipes can only be used by the process that created it and its descendants, while shared memory allows multiple processes to use the same memory region independent of ancestor process which provides a better usability/flexibility. On the other hand, the synchronization is an issue for shared memory unlike pipes and named pipes. The operating system handles the synchronization issues for both pipes. So, it could be said that both pipes are more reliable than shared memory, but less efficient, since the data traveling through the pipe moves through the kernel.

Another difference is that both pipes are built to achieve one-to-one or many-to-one communication in comparison with the shared memory, which is designed to provide many-to-many communication. Besides, the shared memory provides the flexibility of resource management in contrast with pipes, in which it is handled by OS as well.

However, all these flexibilities for shared memory comes along with the burden of heavy and complex programming. Synchronization issues should be considered through utilizing semaphores while programming shared memory. Besides memory management issues should also be addressed and handled by programmer instead of OS.

The other IPC method is message queue. To compare the two IPC methods, the means they employ should be clarified. In shared memory, two processes should be organized such as, the first process should create a shared memory region in its own address space and the second one should attach itself to that memory region of process 1. Here, no explicit kernel intervention takes place, and the processes directly shares first process' allocated memory region which makes the shared memory method quite fast. On the other hand, the message queue utilizes the kernel (system calls), easing the burden of synchronization and thus becoming more reliable, in cost of slowing the message passing process. In message queue, first process passes the message package to the queue and the second one gets the package from kernel, rendering the communication slower.

Despite of the synchronization issues of shared memory, it is still relatively simple to program with respect to message queue, since the message queue is difficult to program for especially irregular, dynamic programs. The last disadvantage of shared memory over message queue is

that it is not suitable for multiple machines, and this is an important usability drawback for current computing systems.

As most of the modern systems now facilitates distributed computing concepts and the more suitable IPC method is message queue, most of the issues related to security could be addressed to message queues. Yet still attention should be paid for shared memory, since wrong permission settings may create vulnerabilities against exploiters trying to inject malicious codes to the region.