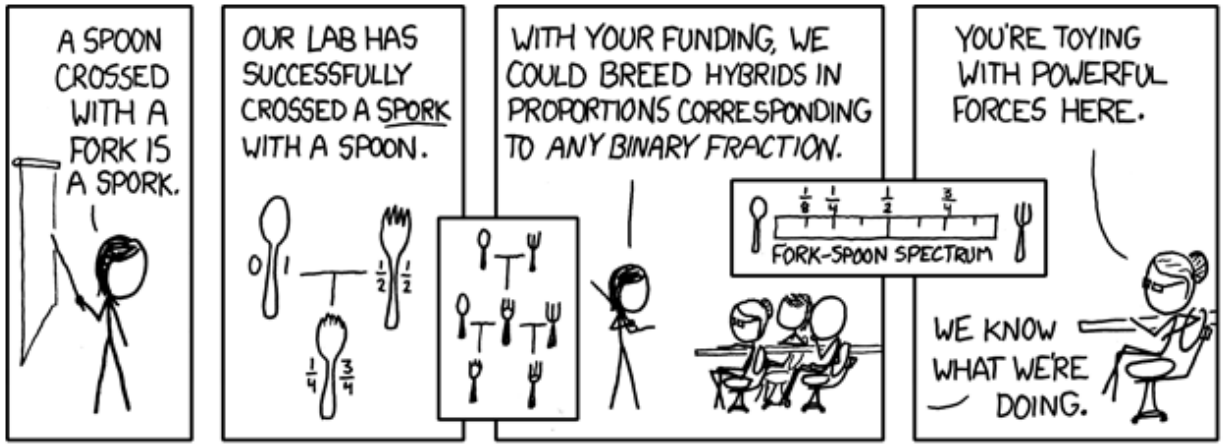
Fork()

Muhammad Asavir / MA6422

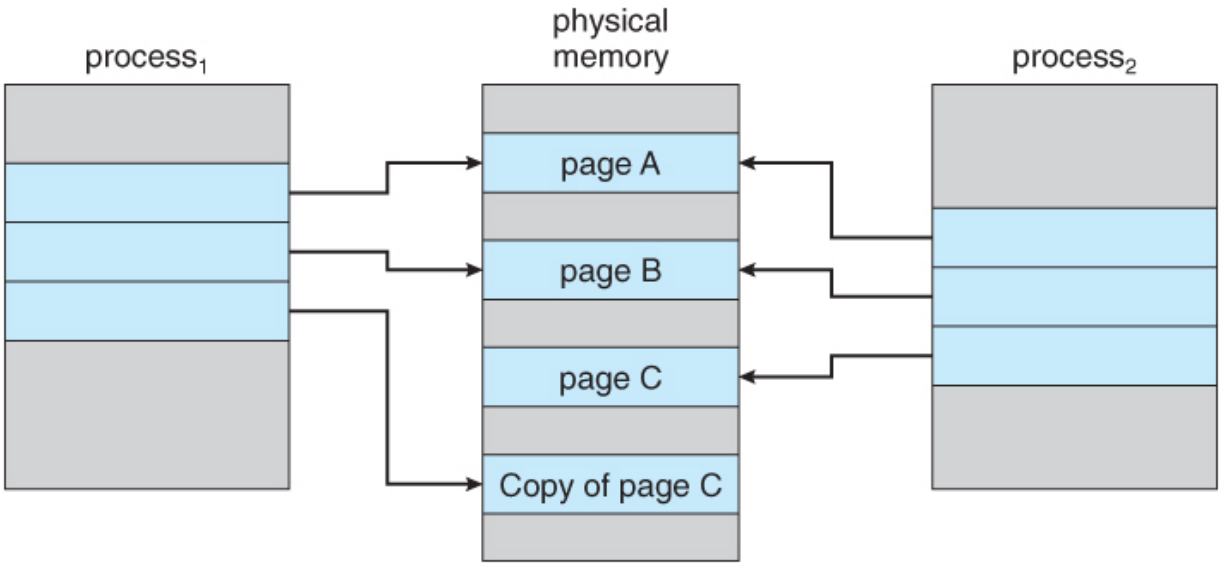


(xkcd)

The UNIX fork() command is an important part of not only UNIX in general but also of network programming in the past few decades. It can be considered a system call and it takes no arguments and returns a system ID, which is of type pid_t and is generally some sort of signed integer. The function will create a new process which is a child process of the original caller. This function is essentially duplicating the calling process. In doing so it continues execution from the point of call in both processes. The parent process ends up getting a value of the child process ID as the return value, whereas the child process will get a value of zero.

The interesting portions of fork() are what happens after it is called, at the OS level, as well as how all the various pieces of memory interact and work with each other. Instead of having a true exact logical copy of the entire process including its memory duplicated, the OS instead shares the memory using copy-on-write. What this entails is that any unmodified memory of the processes remains shared, allowing for only changed portions of memory having their relevant modified pages having a new copy. This method allows for better performance compared to running two full versions in memory. There are a few things that are duplicated however, and these include but are not limited to file descriptors, open directory streams and catalog descriptors. Lastly a couple of values relating to time, such as tms_stime are set back to zero.

An example of the copy-on-write can be seen below:



One neat thing that is not shared is threads in a multi-threaded program. This results in having issues because the child process will only inherit the single thread, and causes many issues with locks and mutexes, both implicit and explicitly set within the program. The general solution to this problem is calling another execve() in order to replace the current process with a new program, and a newly initialized stack, heap and various data segments. References used:

1. https://csl.mtu.edu/cs4411.ck/www/NOTES/process/fork/create.html
2. https://man7.org/linux/man-pages/man2/fork.2.html
3. https://stackoverflow.com/questions/7455161/what-happens-when-i-call-fork-in-unix
4. https://pubs.opengroup.org/onlinepubs/9699919799/functions/fork.html
5. https://man7.org/linux/man-pages/man2/execve.2.html
6. https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html