fork()

The fork system call can be invoked to create a concurrent child process to an existing parent process. If the system call is accepted, the kernel will create a new process by duplicating the entirety of the parent's PCB information (PC, state, files, memory limits, etc.) and assigning a separate PID and address space. Both processes run independently and continue running from the previous line (which is fork). In order to distinguish the two processes, the parent process running fork() will return the child's PID while the child running fork() will return 0.

exec()

In the view of operating system, exec is used to replace the current process with a new one from an executable file. And it basically replaces the entire content of the current process with a new program, then loads the program into the process space and runs it from the entry point. From the coding part in the above picture, the function of exec () basically loads the content of process two from another executable file and replaces the current process with the new process 2 into the current executable file. In this case, process 2 starts to launch while we are using the command of./process2 to execute process 2 from another executable file. And also, process 2 is a child process of process one while we also check that the value of PID is 0. Thus, the exec () function is often used in sequence to get a new program running as a child of a current process.