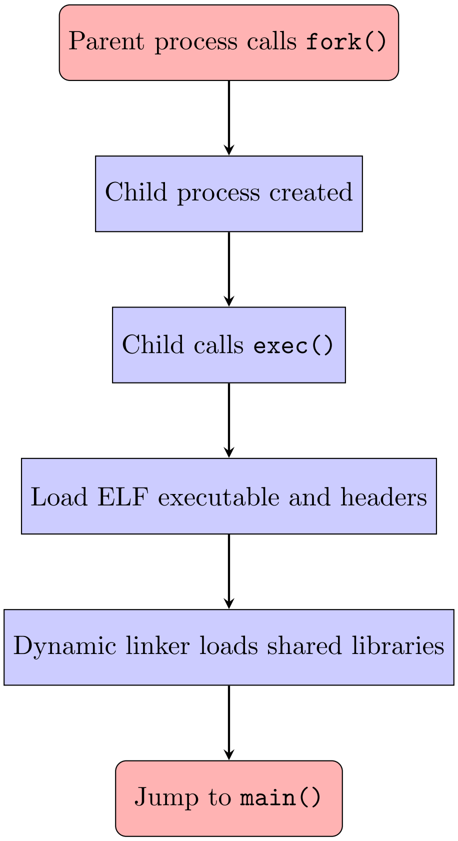
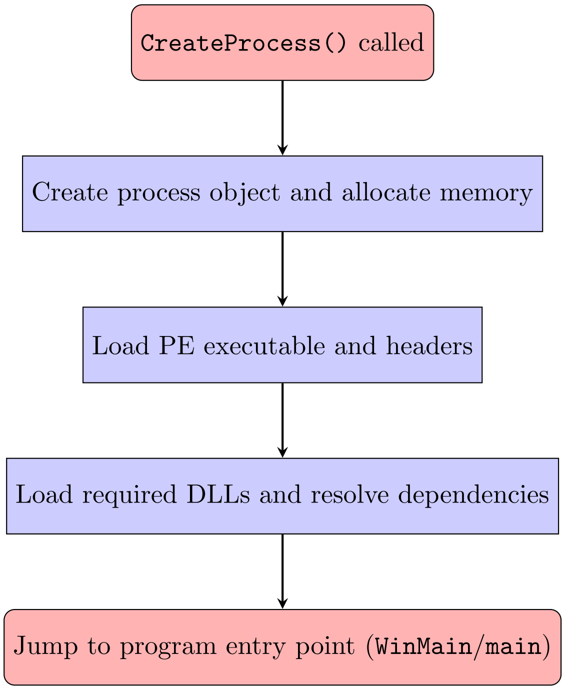
Process Startup in Linux (Similar to RISC-V)

Similar to RISC-V, the process begins with `fork()` which creates a child. This is done so that if there are any errors in the process, it doesn't affect the parent. Moving on, the child calls `exec()` and replaces its memory space with the new executable. All of the program headers and sections are loaded at once. Finally, the linker and loader resolves the shared libraries before the `main` function is invoked.


Parent process calls fork()
Child process created
Child calls exec()
Load ELF executable and headers
Dynamic linker loads shared libraries
Jump to main()

Process Startup in Windows

Similar to Linux, Windows starts a new process with a single call to `CreateProcess()`, which handles both creating the process and loading the program. This call creates a new process object and sets up a virtual memory space for the executable. The PE (Portable Executable) file is loaded, including all its headers and sections. The Windows loader then loads all required DLLs (dynamic link libraries) and resolves dependencies before control is passed to the program's entry point, typically the `WinMain` or `main` function.


CreateProcess() called
Create process object and allocate memory
Load PE executable and headers
Load required DLLs and resolve dependencies
Jump to program entry point (WinMain/main)