The difference between fork(), vfork(), exec() and clone()

123

I was looking to find the difference between these four on Google and I expected there to be a huge amount of information on this, but there really wasn't any solid comparison between the four calls.

I set about trying to compile a kind of basic at-a-glance look at the differences between these system calls and here's what I got. Is all this information correct/am I missing anything important ?

92

`Fork` : The fork call basically makes a duplicate of the current process, identical in almost every way (not everything is copied over, for example, resource limits in some implementations but the idea is to create as close a copy as possible).

The new process (child) gets a different process ID (PID) and has the the PID of the old process (parent) as its parent PID (PPID). Because the two processes are now running exactly the same code, they can tell which is which by the return code of fork - the child gets 0, the parent gets the PID of the child. This is all, of course, assuming the fork call works - if not, no child is created and the parent gets an error code.

`Vfork` : The basic difference between vfork and fork is that when a new process is created with vfork(), the parent process is temporarily suspended, and the child process might borrow the parent's address space. This strange state of affairs continues until the child process either exits, or calls execve(), at which point the parent process continues.

This means that the child process of a vfork() must be careful to avoid unexpectedly modifying variables of the parent process. In particular, the child process must not return from the function containing the vfork() call, and it must not call exit() (if it needs to exit, it should use _exit(); actually, this is also true for the child of a normal fork()).

`Exec` : The exec call is a way to basically replace the entire current process with a new program. It loads the program into the current process space and runs it from the entry point. exec() replaces the current process with a the executable pointed by the function. Control never returns to the original program unless there is an exec() error.

`Clone` : Clone, as fork, creates a new process. Unlike fork, these calls allow the child process to share parts of its execution context with the calling process, such as the memory space, the table of file descriptors, and the table of signal handlers.

When the child process is created with clone, it executes the function application fn(arg). (This differs from for, where execution continues in the child from the point of the fork call.) The fn argument is a pointer to a function that is called by the child process at the beginning of its execution. The arg argument is passed to the fn function.

When the fn(arg) function application returns, the child process terminates. The integer returned by fn is the exit code for the child process. The child process may also terminate explicitly by calling exit(2) or after receiving a fatal signal.

Information gotten form :

- Differences between exec and fork
- http://www.allinterview.com/showanswers/59616.html
- http://www.unixguide.net/unix/programming/1.1.2.shtml

- http://linux.about.com/library/cmd/blcmdl2_clone.htm

Thanks for taking the time to read this ! :)