12.4 • Process Management 475

**System Call: fork (void)**

**fork () causes a process to duplicate. The child process is an almost-exact duplicate of the original parent process; it inherits a copy of its parent's code, data, stack, open file descriptors, and signal table. However, the parent and child have different process ID numbers and parent process ID numbers.**

**If fork () succeeds, it returns the PID of the child to the parent process, and returns 0 to the child process. If it fails, it returns -l to the parent process, and no child is created.**

Figure 12—33 Description of the fork () system call.

fork () is a strange system call, because one process (the original) calls it, but two processes (the original and its child) return from it. Both processes continue to run the same code concurrently, but they have completely separate stack and data spaces.

This reminds me of a great sci-fi story I read once, about a man who comes across a fascinating booth at a circus. The vendor at the booth tells the man that the booth is a matter-replicator; anyone who walks through the booth is duplicated. The original person walks out of the booth unharmed, but the duplicate person walks out onto the surface of Mars as a slave of the Martian construction crews. The vendor then tells the man that he'll be given a million dollars if he allows himself to be replicated, and he agrees. He happily walks through the machine, looking forward to collecting the million dollars ... and walks out onto the surface of Mars. Meanwhile, back on Earth, his duplicate is walking off with a stash of cash. The question is this: If you came across the booth, what would you do?

Linux adds a clone () system call that is, for most uses, the same as fork (). clone () provides for parts of the execution context to be shared (rather than merely copied as with fork () ) between parent and child.

**A process may obtain its own process ID and parent process ID numbers by using the getpid () and getppid () system calls, respectively (Figure 12—34).**

**System Call: getpid (void) getppid (void)**

**getpid () and getppid () return a process's ID and parent process's ID numbers, respectively. They always succeed.** The parent process ID number of PID 1 is 1.