Системный вызов fork() должен предоставить процессу-потомку логически идентичную копию адресного пространства его родителя. В большинстве случаев потомок заменяет предоставленное адресное пространство, так как сразу же после выполнения fork вызывает exec или exit. Таким образом, создание копии адресного пространства (так, как это реализовано в первых системах UNIX) является не оптимальной процедурой.
Вышеописанная проблема была решена двумя различными способами. Сначала был разработан метод копирования при записи (**copy-on-write**). При таком подходе: - страницы данных и стека родителя временно получают атрибут «только для чтения» и маркируются как «копируемые при записи»; - потомок получает собственные карты трансляции адресов (таблицы страниц), которые ссылаются на страницы процесса-предка, т.е. процесс-потомок использует одни и те же страницы памяти вместе со своим родительским процессом; - если кто-то из них (родитель или потомок) попытается изменить страницу памяти, произойдет ошибочная исключительная ситуация по правам доступа, так как страницы доступны только для чтения. Затем ядро системы запустит обработчик исключительной ситуации, который обнаружит, что страница помечена как «копируемая при записи», и создаст новую ее новую копию, которую уже можно изменять. Таким образом, происходит копирование только тех страниц памяти, которые требуется изменять, а не всего адресного пространства целиком.
Если потомок вызовет exec() или exit(), то защита страниц памяти вновь станет обычной, и флаг «копирования при записи» будет сброшен. В системе BSD UNIX представлен несколько иной подход к решению проблемы, реализованный в новом системном вызове vfork(). Функция **vfork**() не производит копирования. Вместо этого процесс-родитель предоставляет свое адресное пространство потомку (потомок получает карты трансляции адресов предка) и блокируется до тех пор, пока тот не вернет его. Затем происходит выполнение потомка в адресном пространстве родительского процесса до того времени, пока не будет произведен вызов exec или exit(), после чего ядро вернет родителю его адресное пространство и выведет его из состояния сна. Системный вызов vfork() выполняется очень быстро, так как не копирует даже карты адресации. Адресное пространство передается потомку простым копированием регистров карты адресации. Однако следует отметить, что вызов vfork является достаточно опасным, так как позволяет одному процессу использовать и даже изменять адресное пространство другого процесса. Это свойство vfork используют различные программы, такие как csh. Программист может воспользоваться vfork() вместо fork(), если планирует вслед за ним сразу вызвать exec().