1. Какова структура таблиц открытых файлов, файлов и описателей файлов после создания процесса?

таблица описателей файлов = 0
таблица файлов = 0
таблица открытых файлов процесса = 3 (т.к три стандартных потока)

2. Для чего используются сигналы в ОС UNIX?

Сигналы – это программное средство, с помощью которого может быть прервано функционирование процесса в ОС UNIX. Механизм сигналов позволяет процессам реагировать на различные события, которые могут произойти в ходе функционирования процесса внутри него самого или во внешнем мире. Каждому сигналу ставятся в соответствие номер сигнала и строковая константа, используемая для осмысленной идентификации сигнала.

3. Какие виды сигналов существуют в ОС UNIX?

Количество стандартных сигналов зависит от версии системы. Ниже приведен перечень, используемый в большинстве диалектов ОС.

SIGABRT	Сигнал генерируется вызовом функции *abort( )* и вызывает аварийное завершение процесса.
SIGALARM	Сигнал посылается по истечении времени, установленного функцией *alarm( )*.
SIGBUS	Сигнал возникает при сбое оборудования.
SIGCONT	Сигнал позволяет возобновить выполнение процесса, прерванного по сигналу **SIGSTOP**.
SIGFPE	Ошибка при выполнении арифметической операции с действительными числами.
SIGILL	Неверная инструкция процессора при выполнении программы.
SIGINT	Сигнал возникает при вводе с терминала <Ctrl+C>.
SIGIO	Завершение асинхронной операции ввода-вывода.
SIGIOT	Ошибочное завершение асинхронной операции ввода-вывода.
SIGKILL	Немедленно завершить процесс.
SIGPIPE	Ошибки при записи в канал межпроцессного ввода-вывода (pipe) (см. 1.7.2.).
SIGSEGV	Ошибка при использовании процессом неверного адреса.
SIGSTOP	Приостановить выполнение процесса.
SIGSYS	Ошибка при выполнении системного вызова.
SIGTERM	Завершить процесс.
SIGTRAP	Аппаратная ошибка при выполнении процесса.
SIGTSTP	Ввод с клавиатуры <Ctrl+Z> (приостановить процесс).

Программа, получив сигнал, может обработать его одним из трех способов:

1. Процесс может проигнорировать сигнал. Это невозможно только для SIGKIL и SIGSTOP. Эти два сигнала позволяют администратору системы UNIX прерывать или приостанавливать процессы в случае необходимости.
2. Процесс может зарегистрировать собственную функцию обработки сигнала. Рекомендуется, по возможности, обрабатывать сигналы, приводящие к преждевременному завершению программы.
3. Если не задана функция обработки сигнала, ядро системы выполняет действия, предусмотренные для его обработки по умолчанию. Если пришел сигнал, связанный с программными и аппаратными ошибками, процесс, как правило, завершается с созданием в текущей директории файла с именем "core". В последний помещается содержимое области оперативной памяти, которая была занята задачей в момент прихода сигнала.

4. Для чего используются каналы?

Для обмена данными между процессами.

В ОС UNIX существует специальный вид взаимодействия между процессами – программный канал. Программный канал создается с помощью системного вызова pipe(), формат которого
int fd[2];
pipe(fd);
Системный вызов pipe() возвращает два дескриптора файла: один для записи данных в канал, другой – для чтения. После этого все операции передачи данных выполняются с помощью системных вызовов ввода-вывода read/write. При этом система ввода-вывода обеспечивает приостановку процессов, если канал заполнен (при записи) или пуст (при чтении). Таких программных каналов процесс может установить несколько. Установление связи через программный канал опирается на наследование файлов. Взаимодействующие процессы должны быть родственными.

5. Какие требования предъявляются к процессам, чтобы они могли осуществлять обмен данными посредством каналов?

Установление связи через программный канал
опирается на наследование файлов. Взаимодействующие процессы должны быть родственными.

6. Каков максимальный размер программного канала и почему?

Важно заметить, что на практике размер буфера канала конечен. Другими словами, только определенное число байтов может находиться в канале, прежде чем следующий вызов fdwrite будет заблокирован. Минимальный размер, определенный стандартом POSIX, равен 512 байтам. В большинстве существующих систем это значение намного больше.

При программировании важно знать максимальный размер канала для системы, так Как он влияет на оба вызова fdwrite и fdread. Он зависит от файловой системы.