3.9)

The current process counter and stack state of the current process must be saved and then the new program counter and stack must be loaded for the kernel to make a context switch.

3.12)

by the end of the loop where will be 16 processes created.

3.15)

when you have one child and one parent process that need to communicate just for the duration of their execution a normal pipe will work perfectly and will automatically be removed upon termination.
When you need a common pipe that will be used by lots of different processes that may not share common ancestors and you want the pipe to exist permanently so it can be uses every time the program runs then a named pipe will need to be used.

3.17)

the child will print 0, -1, -4, -9, -16 and the parent will then print 0, 1, 2, 3, 4 since they both of there own copy of the array nums and they are not using shared memory.

3.18)

a Sychronous is easy to program because the order of events in guarenteed, by asychronous is faster because you can have independant operations running at the same time, the programmer just has to spend time working on making sure events are happening in the right order.

b With automatic buffering allows for infinite length message queues which free's up the sender to work at its own pace. Explicit buffering uses a specific buffer size and ensures there won't be wasted memory due to a massive que building up.

c send by copy allows the receiver to change the message without affecting the sender since they will both have different versions, send by reference allows the receive to view changes made by the sender after it has been received but also saves memory since a copy of the message doesn't need to be made.

d with fixed message sizes there is a know number of messages that can be held by a buffer, there is also less overhead because there doesn't need to be a mechanism to specify the message size. With variable message sizes there is more flexibility in what can be sent and how it is sent for the programmer, but the system has to have a method for knowing each message size and fiting the messages into the buffer.