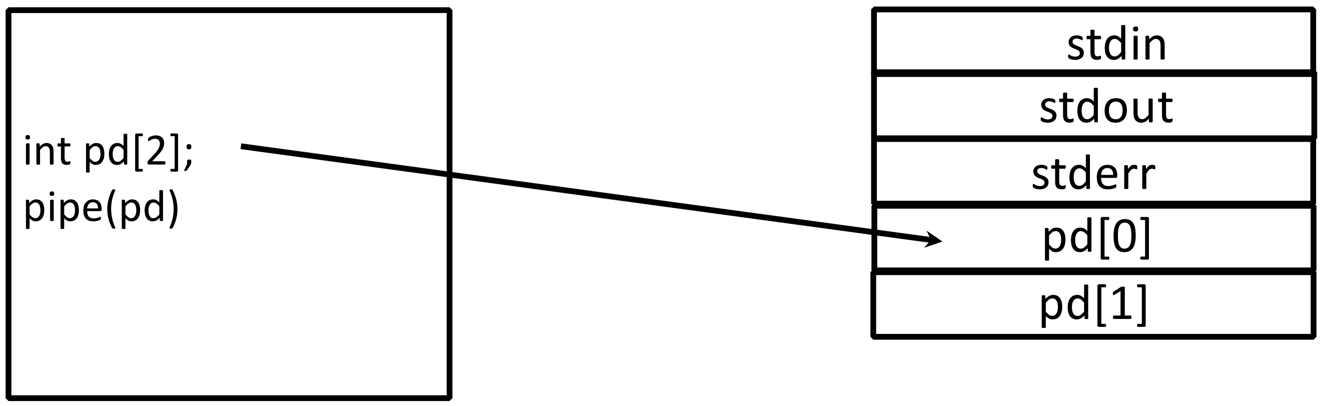
Pipes

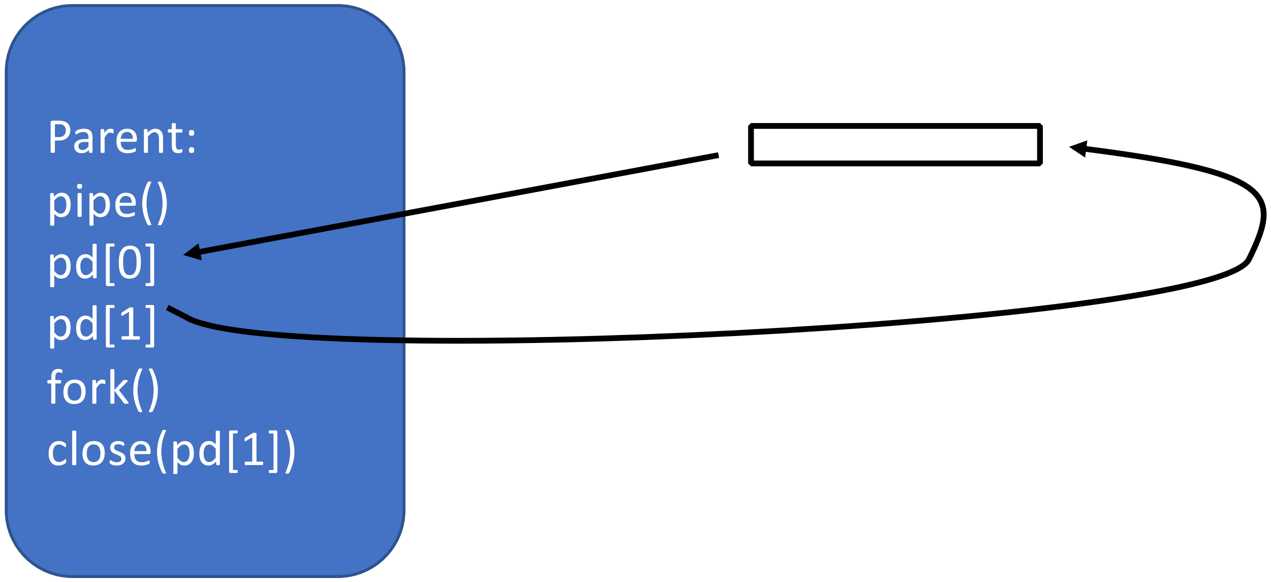
- pipes are created using the system call pipe()
- Like named pipes, pipes are a FIFO byte stream between two processes
- Unlike named pipe, pipes require a common process ancestor
- Any process that writes to a pipe will hang until the data is read by another process.
- Any process that reads from a pipe will hang if there is no data in the pipe AND if there is any process with an open file handle to the pipe

file descriptor table after pipe()

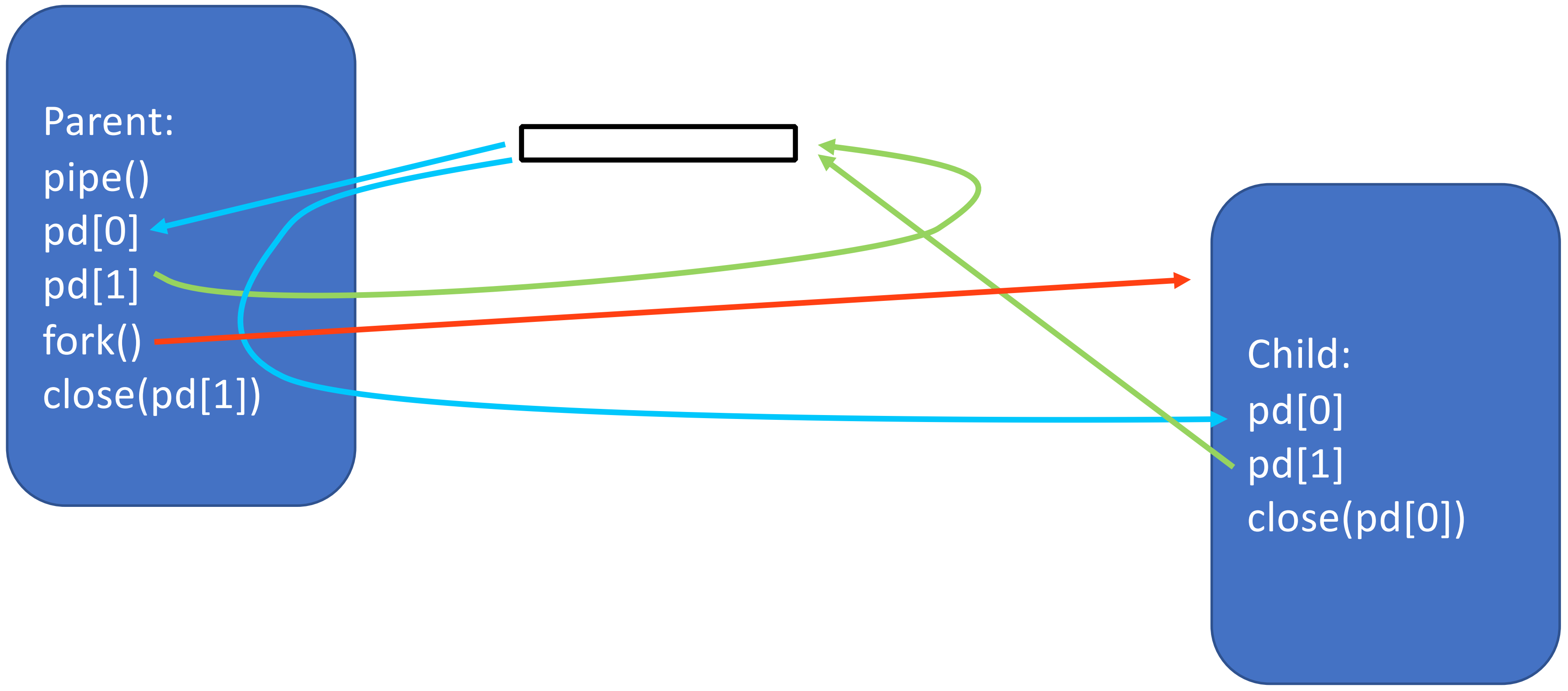
- Pipes are the oldest form of UNIX interprocess communication.
- They have the following limitations.
  - The are unidirectional
  - They are meant to be used between processes that have a common ancestor. ex: between parent and child, or between two children etc.



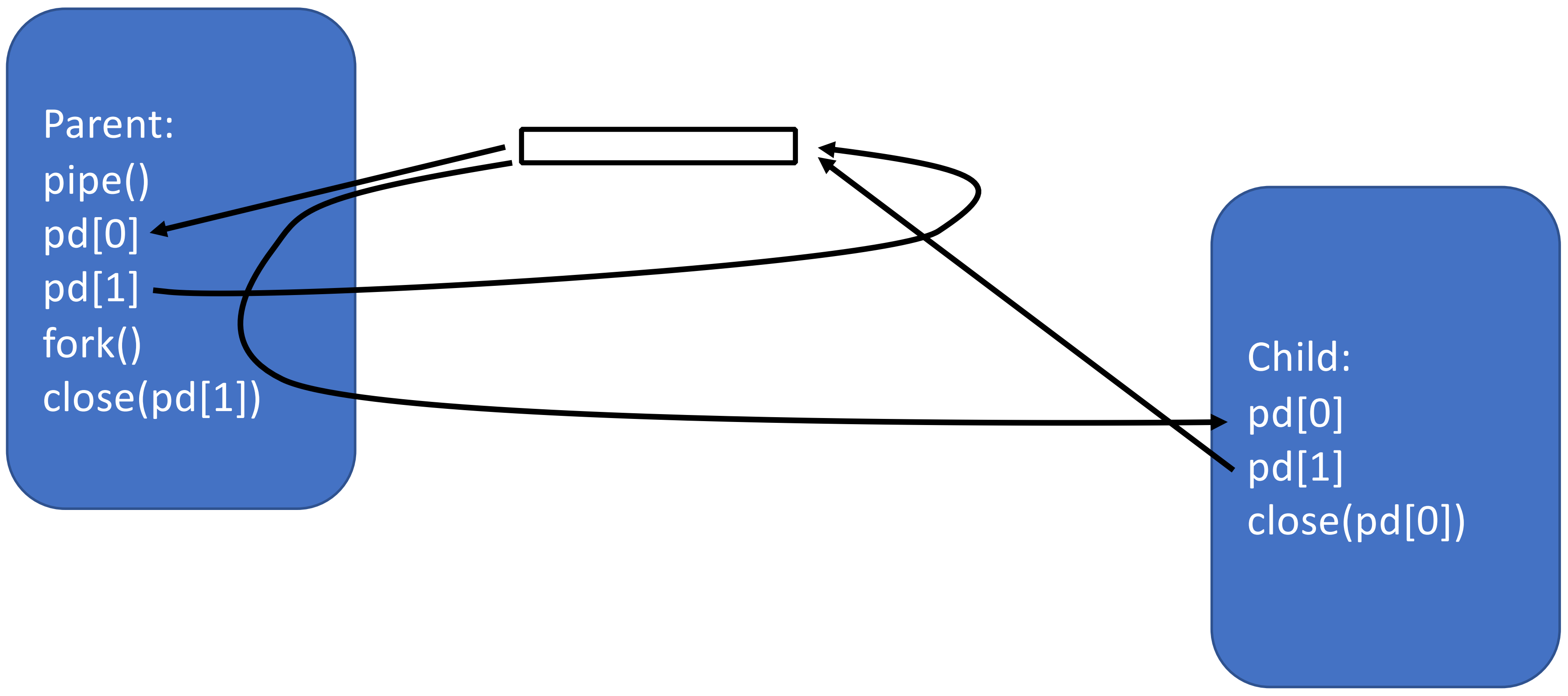
pipe in one process



pipe between two processes - after fork

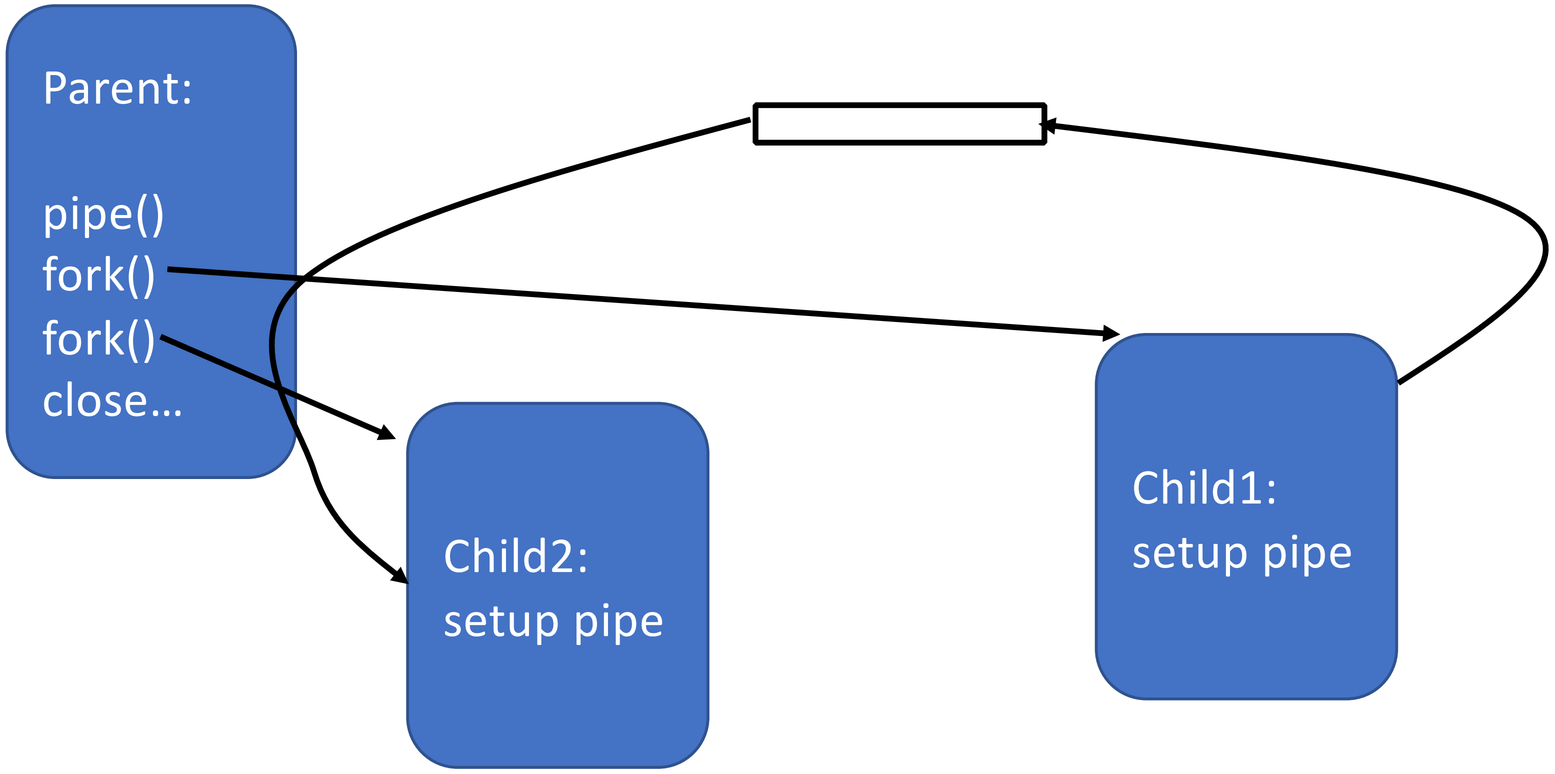


pipe between two processes - setup!



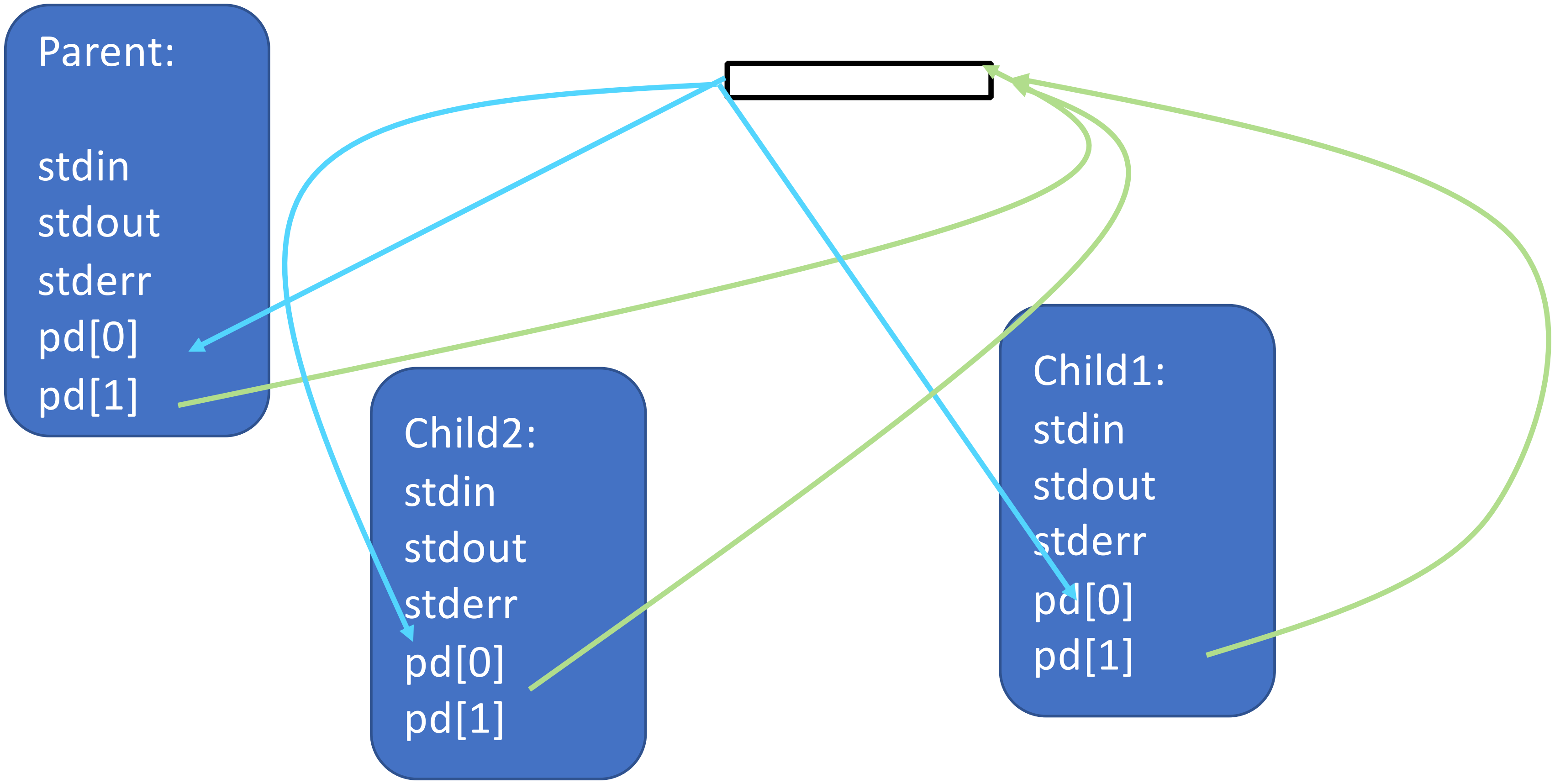
pipe between two commands

- cat /etc/motd | grep upgrade && echo “New software installed”
- Parent can only get exit status of child not grand children.
    - i.e. Parent has to fork both children.



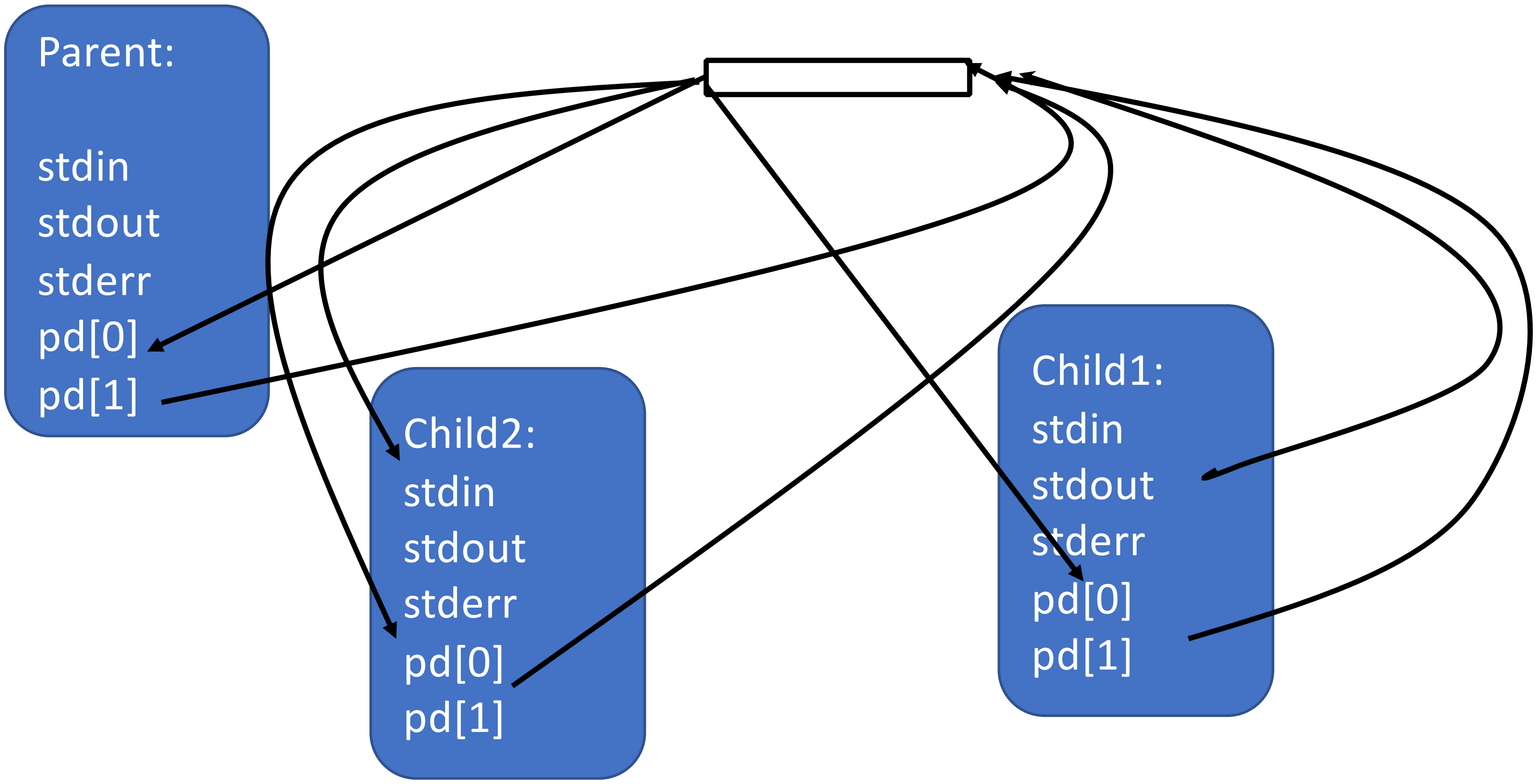
pipe between two commands

- cat /etc/motd | grep upgrade && echo “New software installed”
- Parent can only get exit status of child not grand children.
  - i.e. Parent has to fork both children.



pipe between two commands

- cat /etc/motd | grep upgrade && echo “New software installed”
- Parent can only get exit status of child not grand children.
  - i.e. Parent has to fork both children.


Parent:
stdin
stdout
stderr
pd[0]
pd[1]
Child2:
stdin
stdout
stderr
pd[0]
pd[1]
Child1:
stdin
stdout
stderr
pd[0]
pd[1]