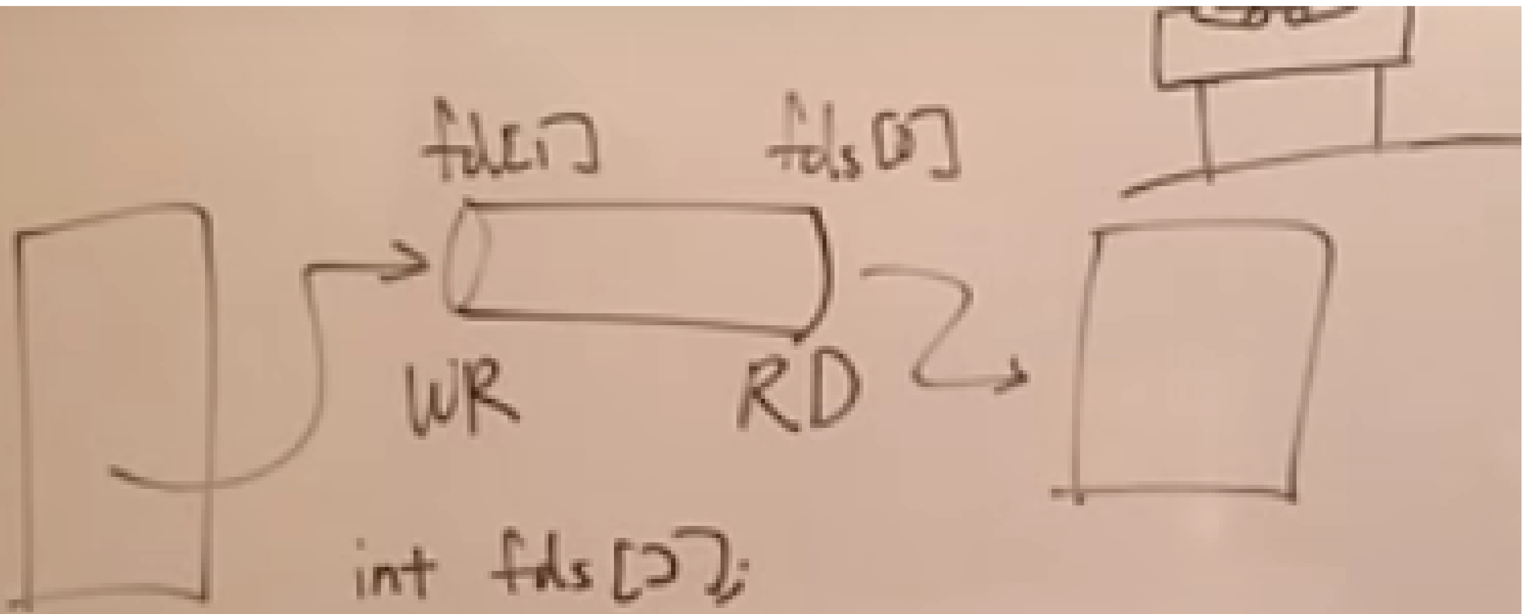
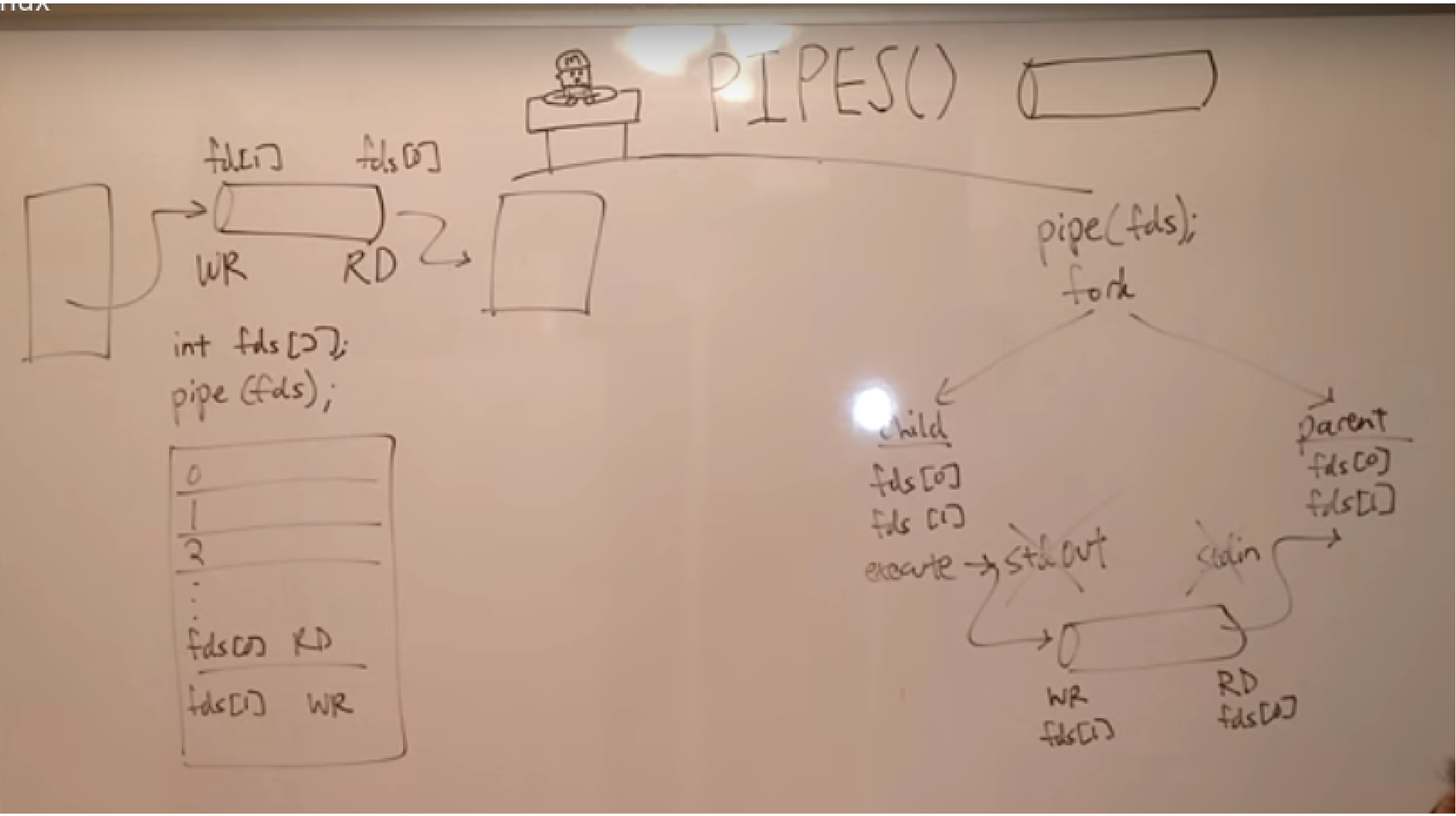


```
int fds[2];
pipe(fds);
```

0	
1	
2	
. . .	
fds[0]	RD
fds[1]	WR


PIPES()
fds[1]
fds[0]
WR
RD
int fds[2];
pipe (fds);
0
1
2
fds[0] RD
fds[1] WR
pipe(fds);
fork
child
fds[0]
fds[1]
execute
~~stdout~~
~~stdin~~
parent
fds[0]
fds[1]
WR
fds[1]
RD
fds[0]

pipe() system call -

This system call is used to create a read-write pipe that may later be used to communicate with a process we'll fork off. The call takes as an argument an array of 2 integers that will be used to save the two file descriptors used to access the pipe.

The first to read from the pipe, and the second to write to the pipe. Here is how to use this function: int fd[2]; if (pipe(fd) < 0) perror(“Error”);

If the call to pipe() succeeded, a pipe will be created, fd[0] will contain the number of its read file descriptor, and fd[1] will contain the number of its write file descriptor.

Our program first call fork() to create a child process. One (the parent process) reads write to the pipe and child process reads the data from the pipe ans then prints the data to the screen.