6.033 One-Pager #2

February 15, 2005
Nada Amin
Joseph, TR 11

UNIX pipes allow two processes to communicate by sending stream of bytes from one to the other. Here, I discuss one significant limitation of pipes: they are one-way communication channels.

Using a two-way pipe, a developer could easily create a new interface to an old program. For example, `dc` and `bc` are command-line calculators with equivalent functionality but different interfaces, as `dc` uses postfix notation and `bc` infix notation. Instead of duplicating the functionality of `dc`, a developer could create `bc` using a two-way pipe, converting the user input into `dc` input and the `dc` output back into `bc` output. We could even imagine a more complex GUI calculator that communicates with a command-line utility for all the calculations. Two-way pipes would allow the developer to create many different interfaces to the same underlying program.

In the same way that standard pipes allow a program to be composed as a chain of simpler programs, two-way pipes would allow a program to be composed as a chain of interactions between simpler programs. We could imagine an AI chatbot constructed from many interacting components: a program that turns an affirmation into a question, one that queries the web to find the meaning of the web, a generator of random quotes, etc.

In conclusion, the one-way limitation of standard pipes significantly restricts the ability of developers to compose programs from connected components. Indeed, many systems propose two-way pipes, known as "stream pipes" to bypass this limitation.