Pipes

COMS10012 Software Tools

standard IO

Unix Philosophy

It is easier to maintain 10 small programs than one large program. Therefore,

1. Each program should do one thing well.
2. Programs should be able to cooperate to perform larger tasks.
3. The universal interface between programs should be a text stream.

source

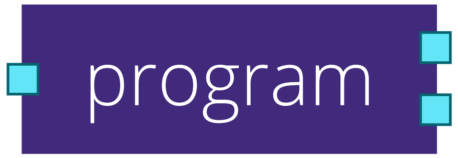
```
#include <stdio.h>
// gives stdin etc.
// fread, fwrite, FILE* - C abstraction

#include <unistd.h>
// pulls in /usr/include/sys/unistd.h
// read, write – POSIX abstraction

#define STDIN_FILENO    0
#define STDOUT_FILENO   1
#define STDERR_FILENO   2
```

standard input/output

Internally, programs read(fd, buffer, size) and write(fd, buffer, size).

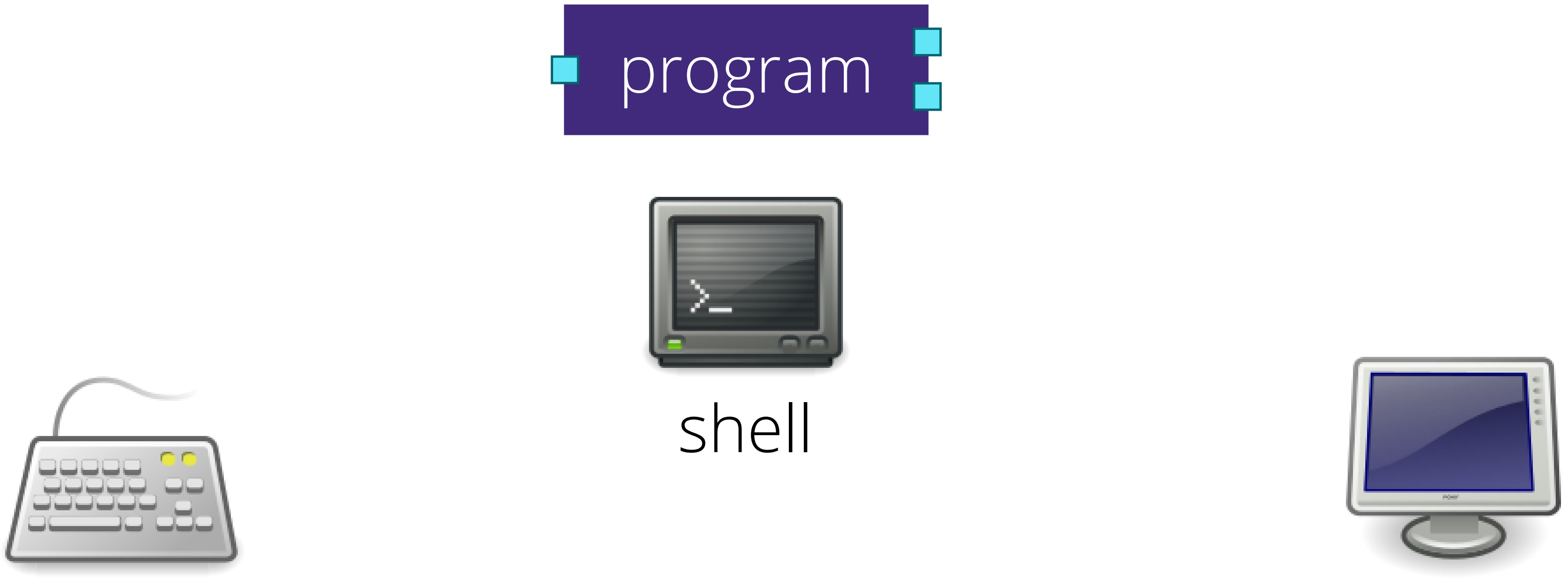
Each program starts with three file descriptors open:

0 = standard input
1 = standard output
2 = standard error



standard input/output

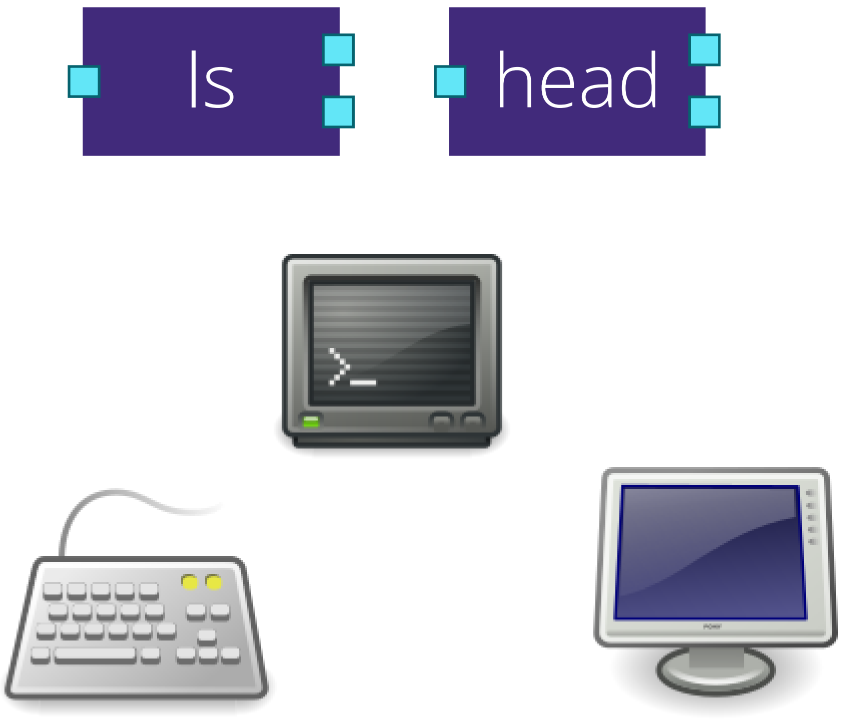
Running a program in the terminal:



pipes

pipe

```
$ ls -1 | head
```



```
head [-n NUM]
tail [-n NUM]
```

pipe

```
$ ls -1 | grep software | sort
```

```
grep: "global regular expression parser"
sort: read all lines into buffer, sort, output
uniq: remove duplicates immediately following
      best used as: command | sort | uniq
```

grep

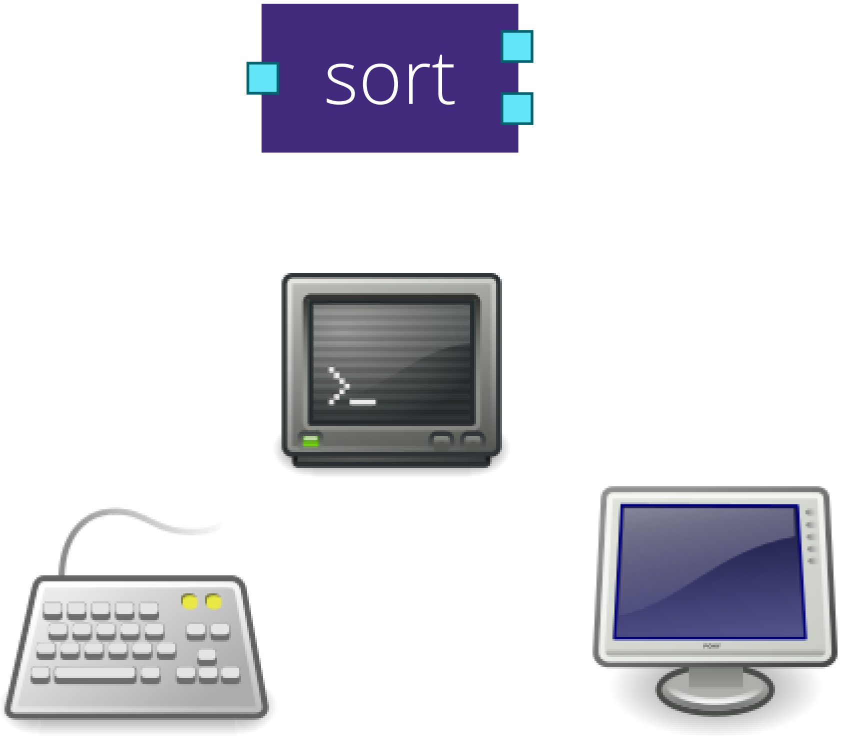
```
$ grep PATTERN FILENAMES
```

```
$ grep -nHi PATTERN FILENAMES
```

```
$ grep [OPTIONS] PATTERN
```

sort

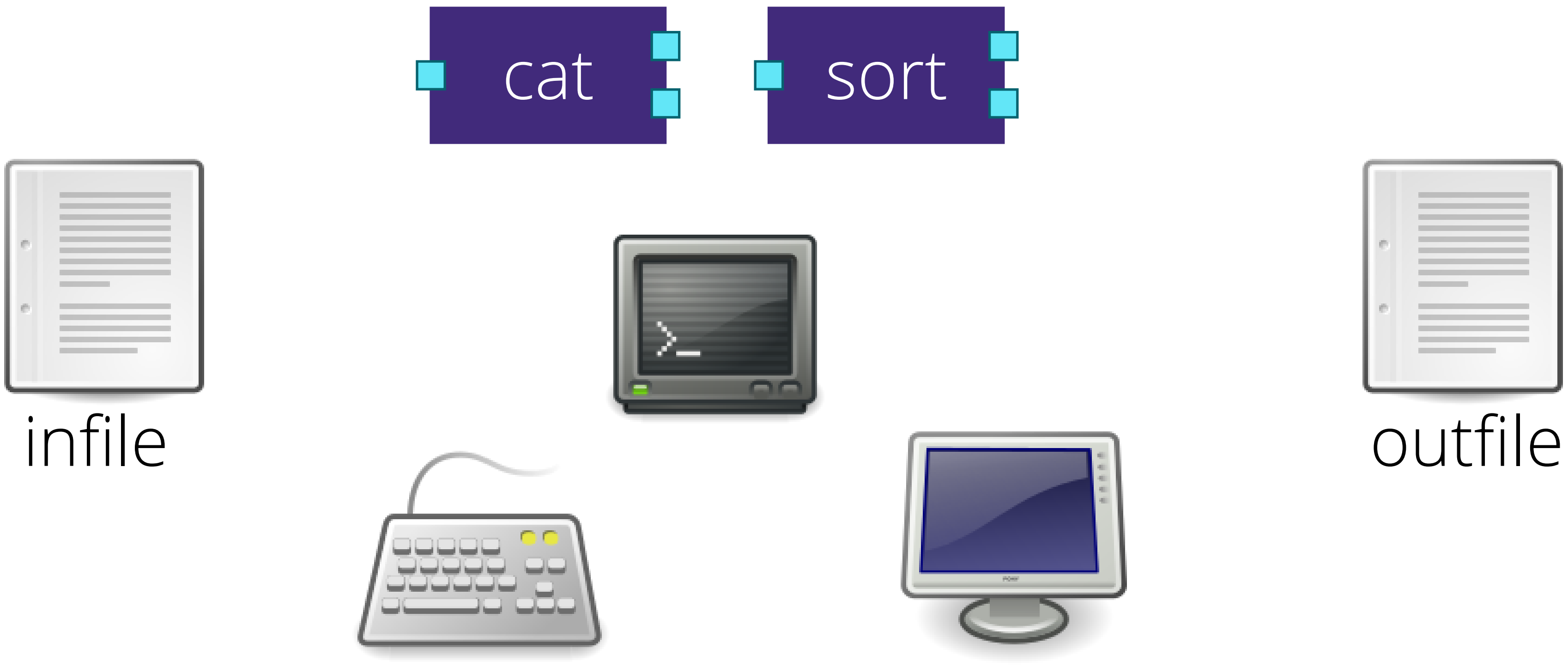
```
$ sort
aaa
ccc
bbb
^D
aaa
bbb
ccc
$
```



redirects

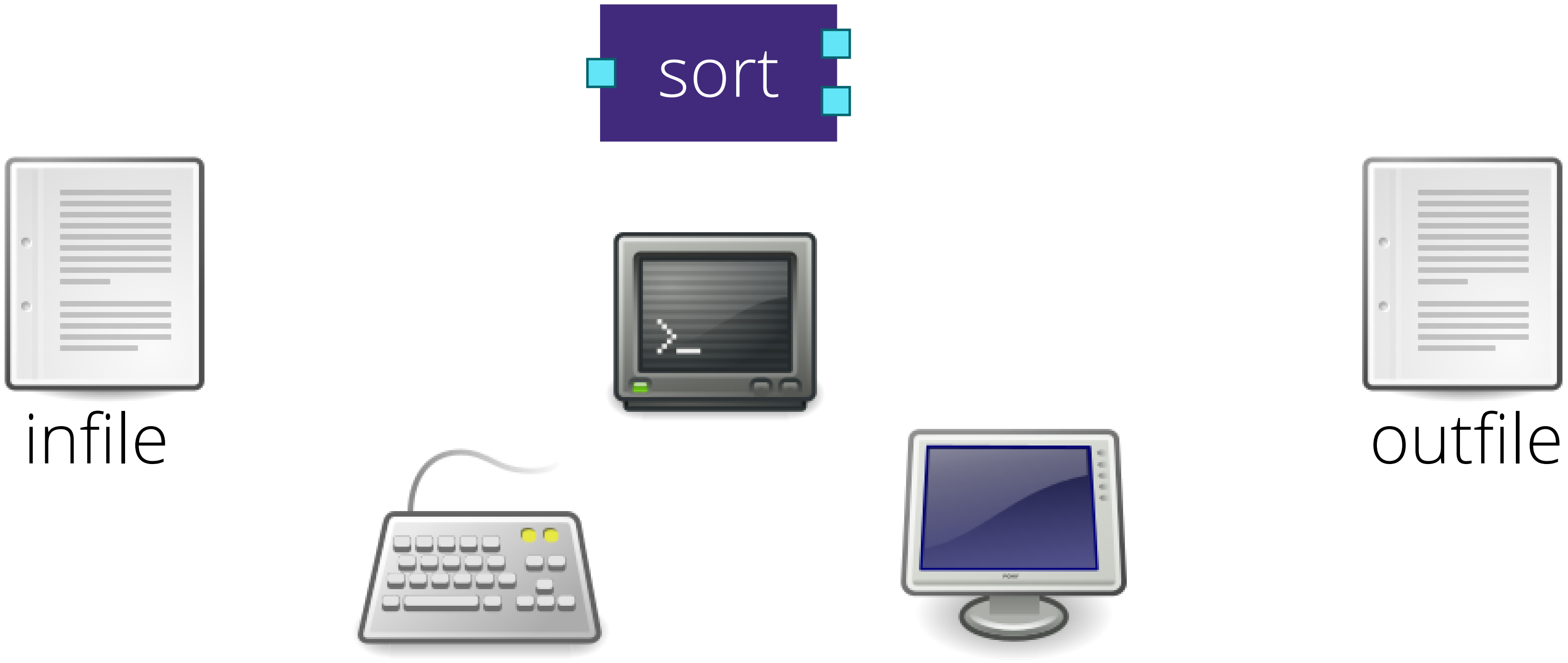
redirect

```
$ cat infile | sort > outfile
```



redirect

```
$ sort < infile > outfile
```



redirect

`$ COMMAND > FILE`	`overwrites FILE`
`$ COMMAND >> FILE`	`appends to FILE`

error redirect

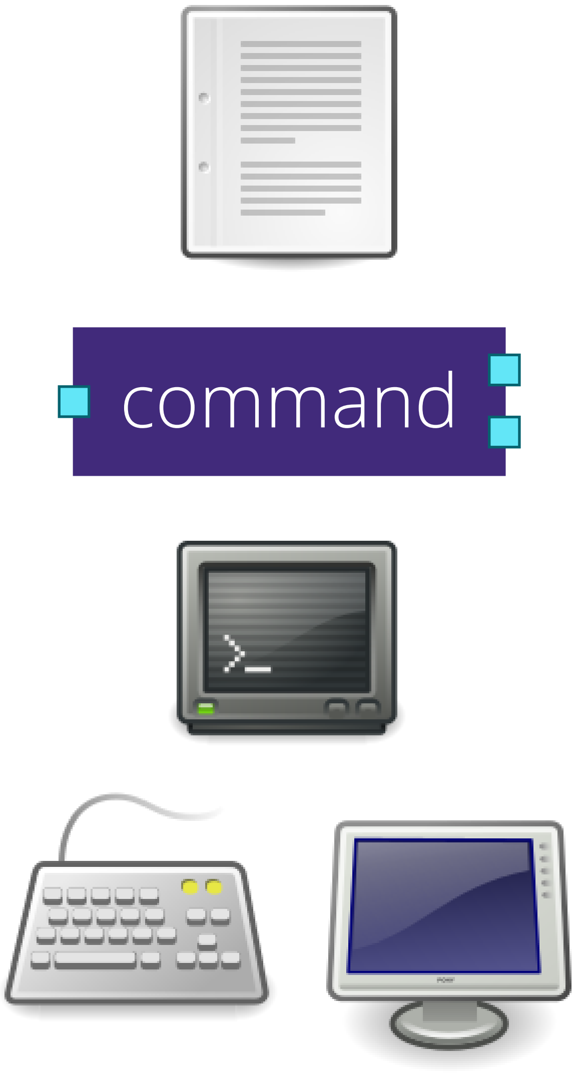
```
$ COMMAND > FILE 2> FILE2
$ COMMAND > FILE 2>&1
```

not:

```
$ COMMAND 2>&1 > FILE
```

ignore output:

```
$ COMMAND > /dev/null
```



files vs streams

A program that uses a standard stream can be told to use a file instead by

- `PROGRAM < FILE` (standard input)
- `PROGRAM > FILE` (standard output)
- `PROGRAM 2> FILE` (standard error)

files vs streams

A program that expects a filename can be told to use standard input/output instead by:

- using the filename **`-`** (single dash), if the program supports it
- using the filename **`/dev/stdin`** etc., if your OS supports it

Filenames with dashes

Filenames starting with dashes are generally considered bad.

If you really want to address one (e.g. you created one by mistake), use e.g.

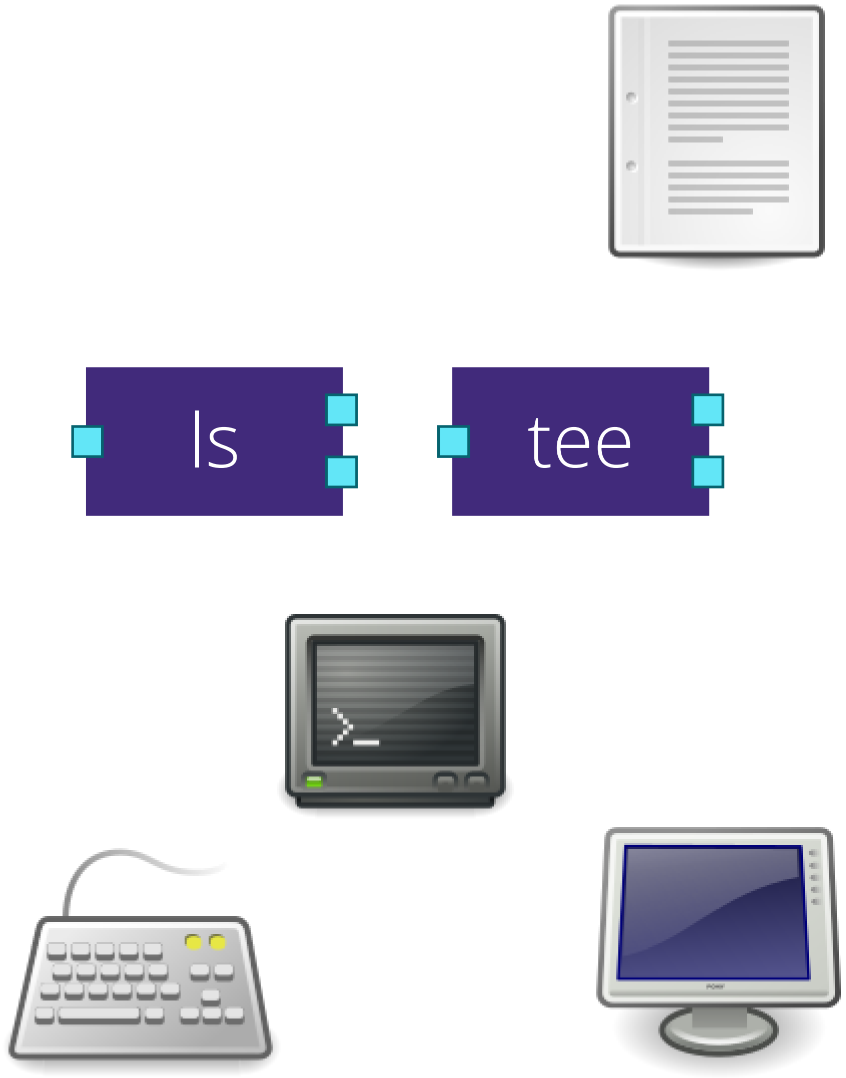
```
$ cat ./-
$ rm ./-f
```

advanced

tee

```
$ ls | tee FILE
```

tee: takes a filename as argument and writes a *copy* of input to it, as well as to stdout



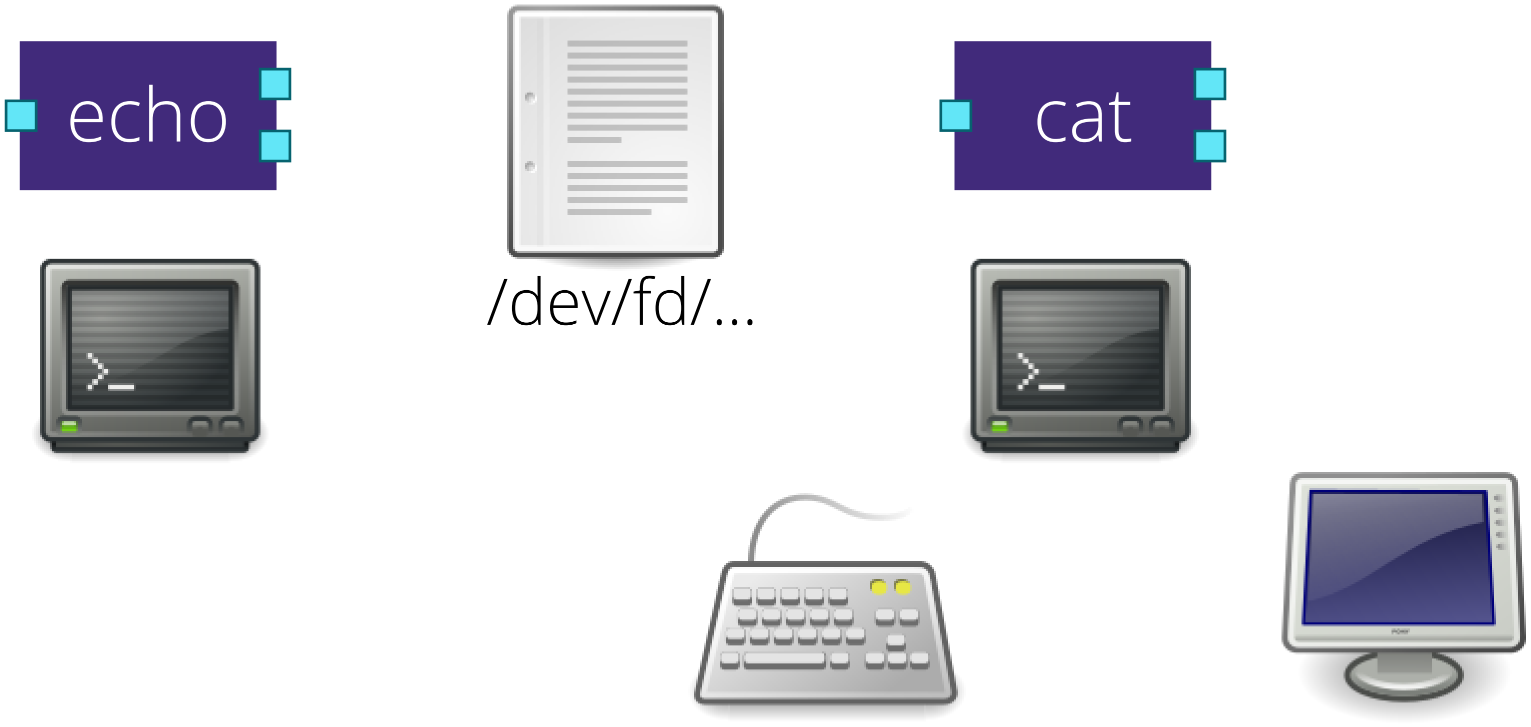
need a file, want a pipe

If PROGRAM wants a file to read from, how can I pipe something in?

```
$ PROGRAM <(SOMETHING)

$ cat <(echo "Hi")
Hi
$ echo <(echo "Hi")
/dev/fd/63
```

subshell

```
$ cat <(echo "Hi")
```



subshell to argument

```
$ COMMAND $(SOMETHING)
```

```
$ echo $(echo Hi | sed -e s/Hi/Hello/)
Hello
```

old-fashioned way, with backticks:

```
$ COMMAND `SOMETHING`
```