Pipes 2

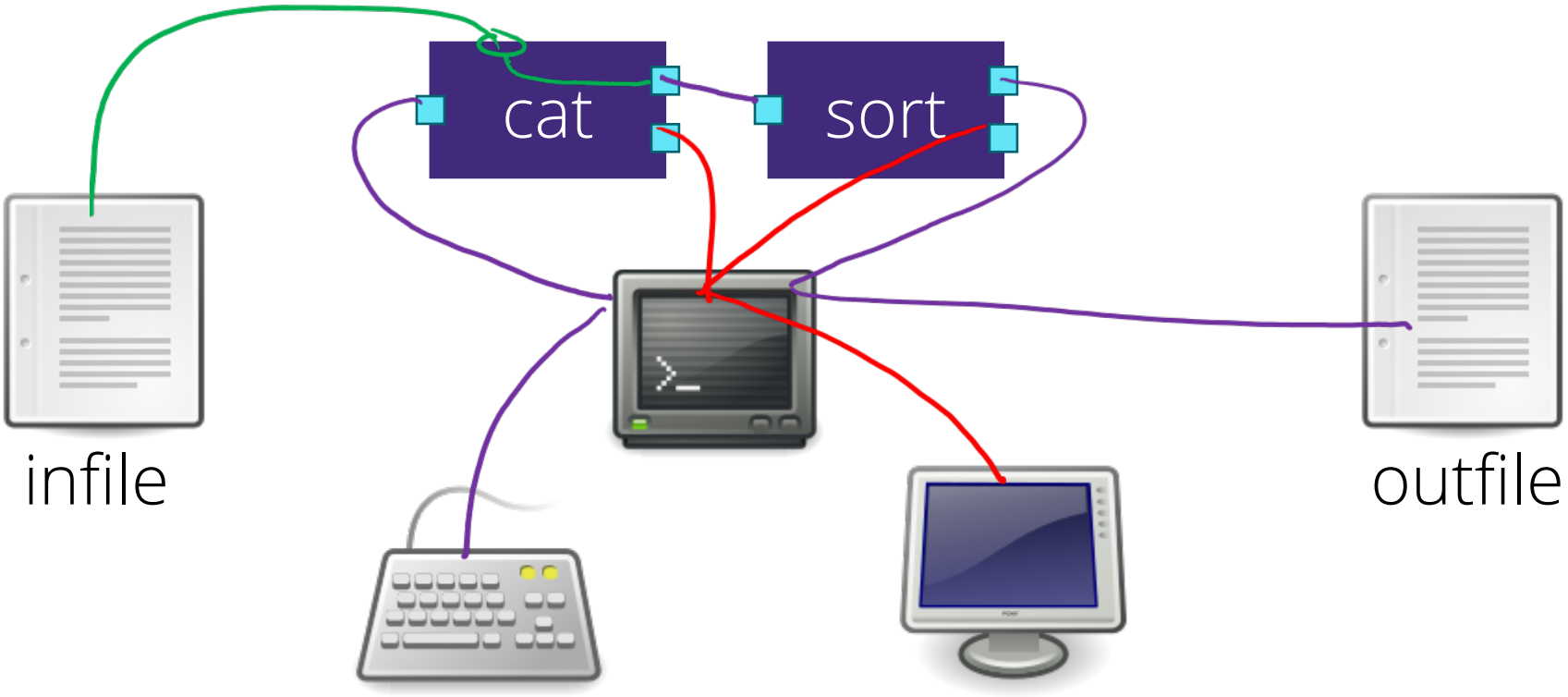
COMS10012 / COMSM0085

Software Tools

redirects

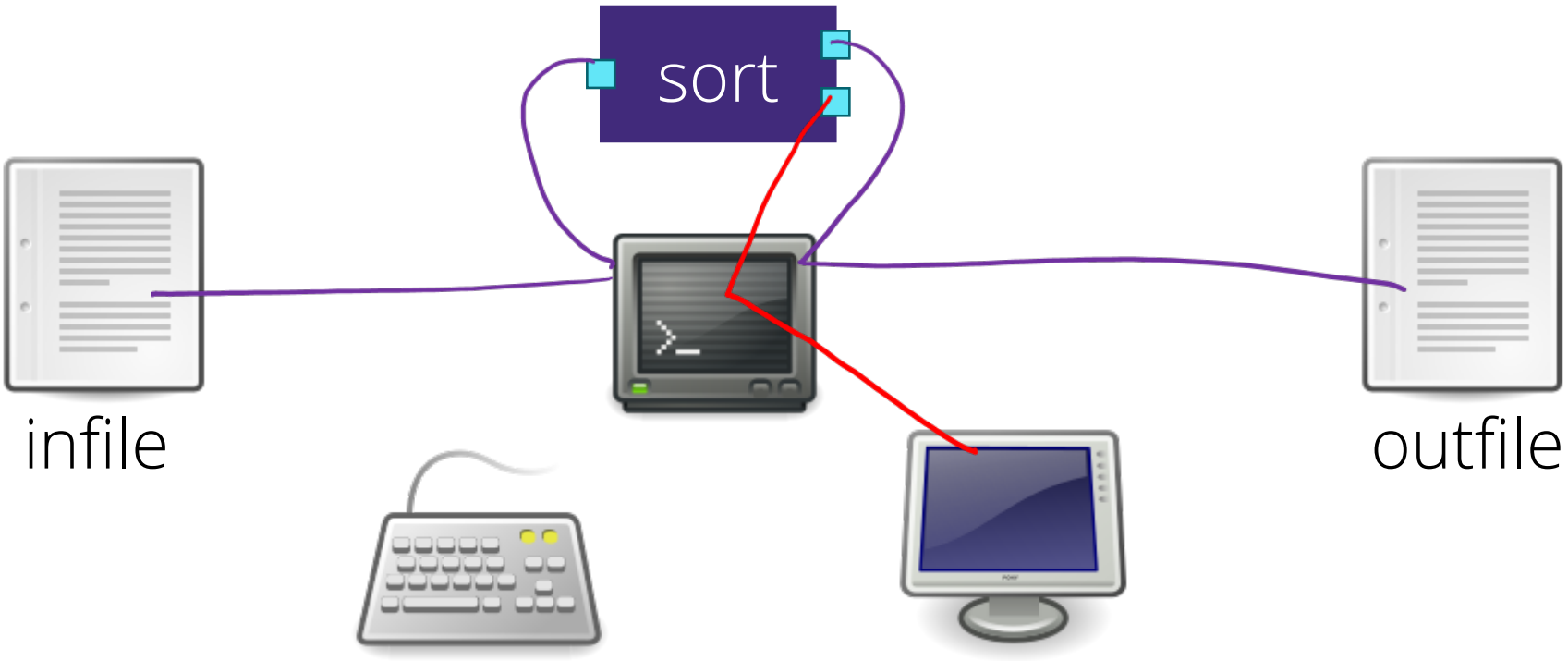
redirect

```
$ 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
```

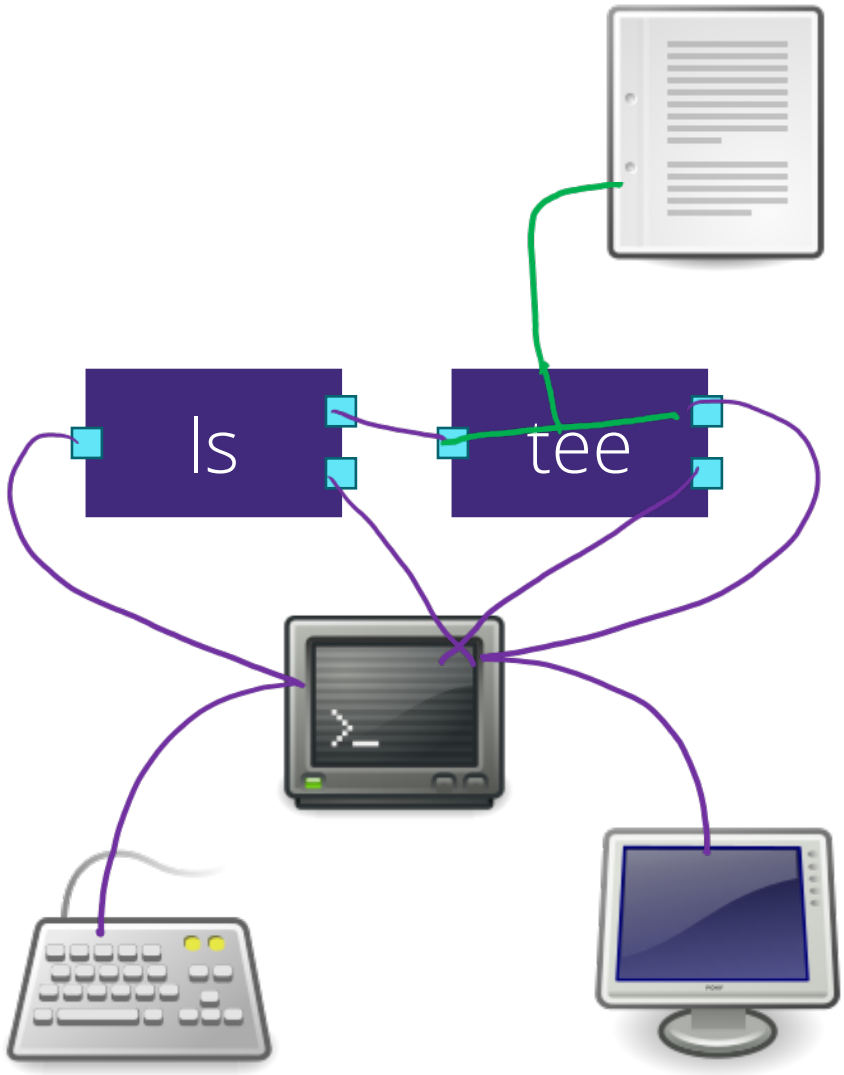
advanced

tee

```
$ ls | tee FILE
```

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

Record logs



pagers

```
$ ls | less
```

`less` is a pager: it displays text on your screen, one page at a time.

- Up/Down arrows scroll,
- Space/Enter advance a page,
- / (forward slash) opens a search,
- q quits.

This takes direct control of the screen (terminal).

sed

```
$ echo "Hello World" | sed -e 's/World/Universe/'
Hello Universe
```

`sed` stands for stream editor – it can change text using a regular expression as it passes from input to output.

`s/ONE/TWO/[g]` replaces the first match for ONE (all matches, with /g) with TWO. Regular expressions are supported.

need a file, want a pipe

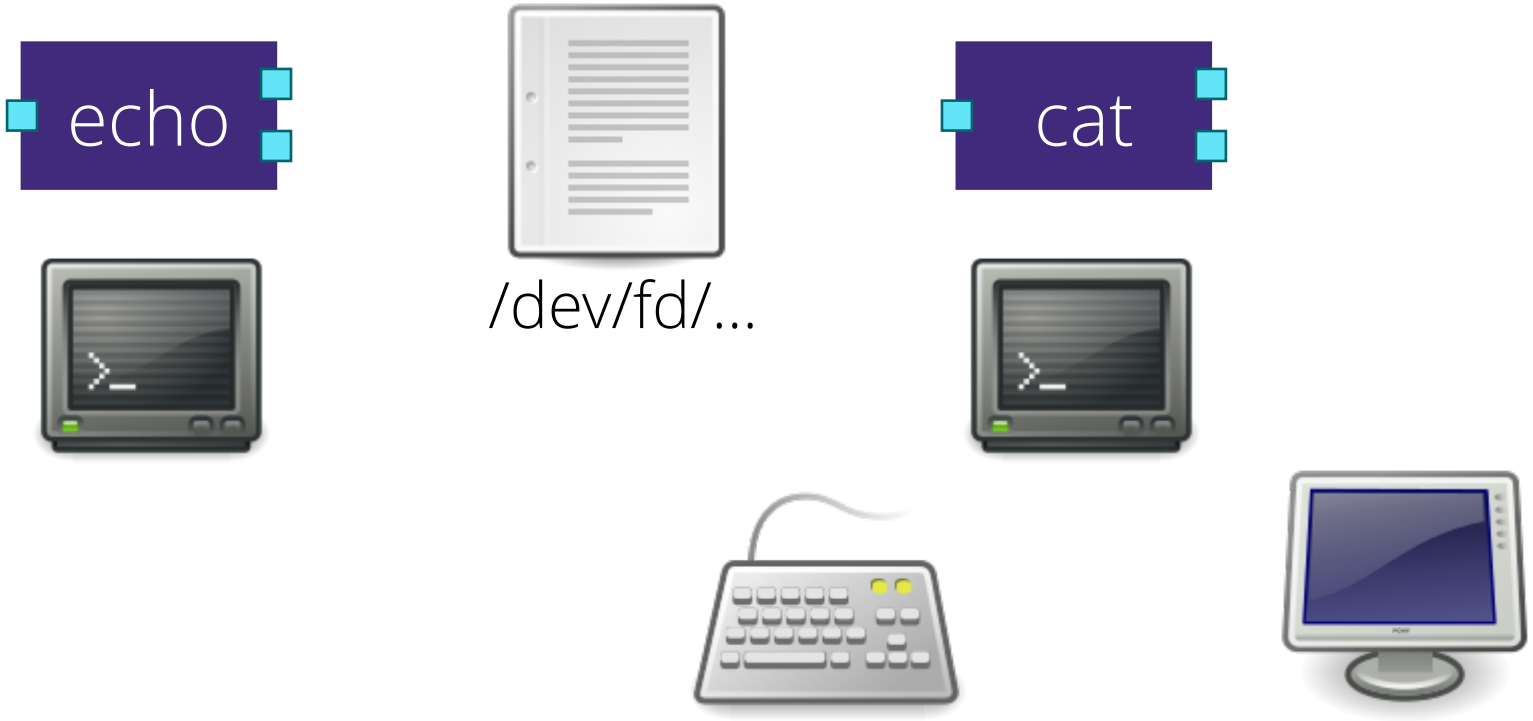
If PROGRAM wants a file to read from, how can I pipe something in?

```
$ 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`
```