Pipes 1

COMS10012 / COMSM0085

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

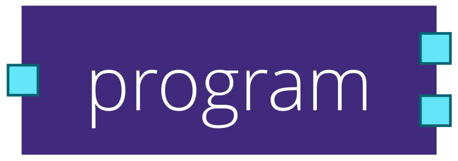
```
#include <stdio.h>
// gives stdin etc.
// fread, fwrite, FILE* - C abstraction

#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).

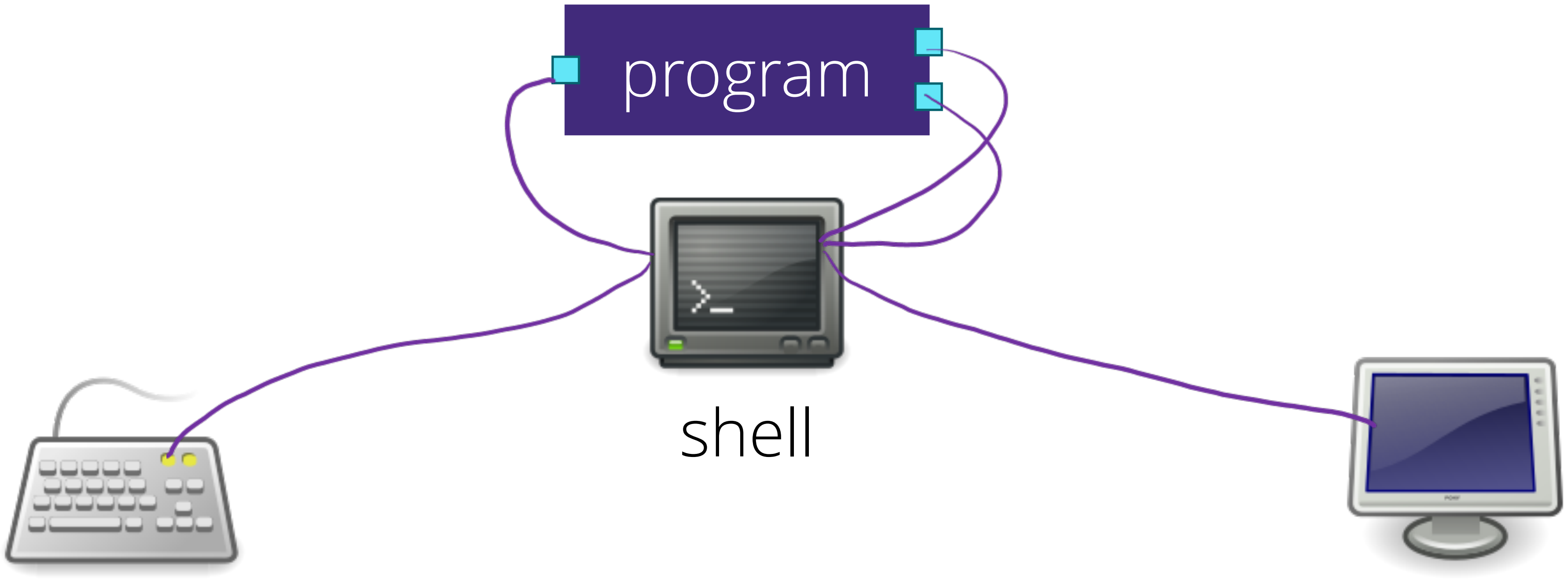
Each program starts with three file descriptors open:

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



standard input/output

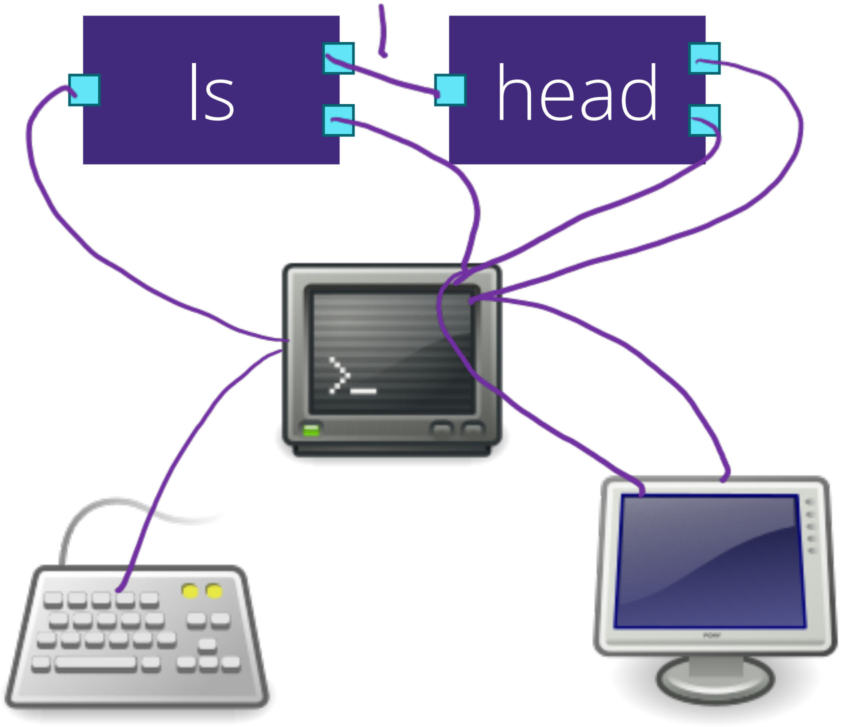
Running a program in the terminal:



pipes

pipe

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



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

pipe

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

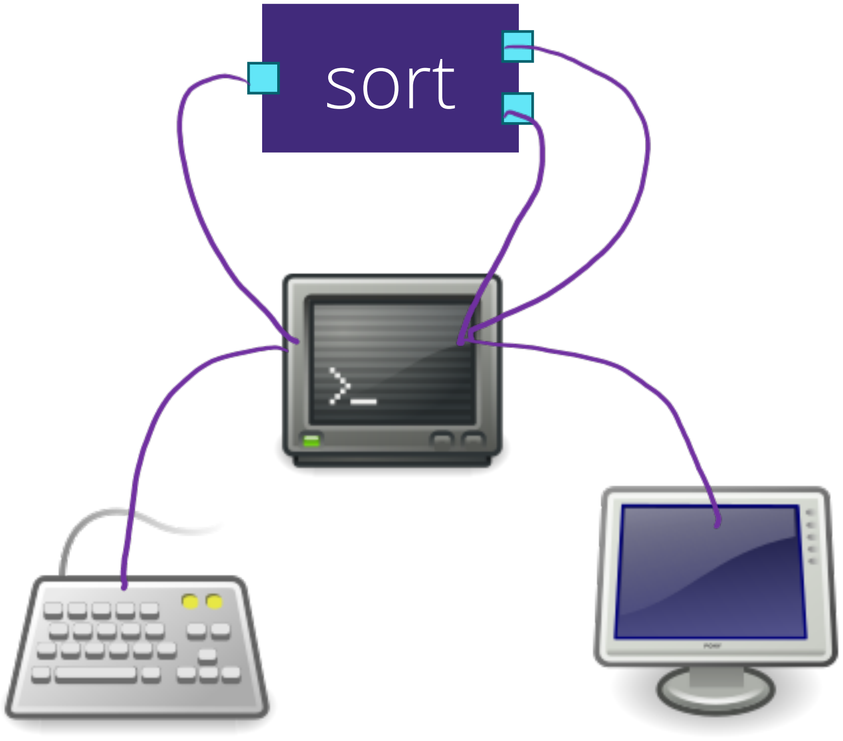
```
grep: "global regular expression parser"
sort: read all lines into buffer, sort,
output
uniq: remove duplicates immediately following
      best used as: command | sort | uniq
```

grep

```
$ grep PATTERN FILENAMES

$ grep -nHi PATTERN FILENAMES

$ grep [OPTIONS] PATTERN
```

sort

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



Exercise 2.2 (document called words (49'000 long))

- The first word in the file. *Can you guess what it will be?* head -1 words
- The last word in the file. *Can you guess this one?* tail -1 words
- The number of words in the words file - there is one word per line. wc -L words
- The 6171st word in the file. *Can you read my mind and guess this word directly?* head -6171 words
- All words containing the letter Q, capitalised. (A regular expression containing a string of one or more letters matches all strings that contain the expression as a substring.) grep 'Q' words
- All words starting with the letter X. The regular expression `X` would match an X anywhere in the word, but `^X` matches an X only at the start of the string. grep '^X' words
- All words ending in j. (The expression `'j$'` matches a j only at the end of the string, but you have to single-quote it to stop the shell from interpreting the dollar sign). grep 'j$' words
- The number of words containing the letter Q, ignoring case (e.g. capitalised or not). grep -i 'Q' words
- The first five words containing the letter sequence `cl`. grep 'cl' words | head -5
- All words containing the sequence "kp", but not "ckp". *Can you guess any of these?* grep 'kp' words | grep -v 'ckp' words
- The last 15 words of exactly two letters. The expression `.` (period) matches a single character, and `'^...$'` for example would match all strings of the format *exactly three characters between start and end of string*. You need to quote it because of the dollar sign. grep '^..$' words | tail -15
- All words from the first 100 words on the list, which contain the letter y. head -100 words | grep -i 'y'
- The first five words that are among the last 100 words on the list, and contain the letter y (whether capitalised or not). tail -100 words | grep ~~-i~~ 'y' | head -5
- All three-letter words with no vowels (aeiou).The regular expression `'[aeiou]'` matches any string that contains one of the bracketed characters; you need quotes to stop the shell from interpreting the brackets. Remember to exclude words with capitalised vowels as well. *There are 343 of these.* grep '^...$' words | grep -iv '[aeiou]' | wc -L
- All words of exactly 7 letters, where the third one is an e and the word ends "-ded". *This kind of search is really useful for crosswords. There are 14 words of this form, can you guess them?*

grep '^..e.ded$' words | wc -L

Bonus regular expression question:

- Find all words that start with a P (whether capitalised or not), and contain at least four instances of the letter a. Putting a `*` after something in a regular expression searches for *any number of repetitions of this, including 0* so for example `'a*'` would find words with any number of the letter a, including 0 (which is not what you want here). You need single quotes to stop the shell from expanding the `*`. *Can you guess the words? There are 14 hits in the solution but essentially five words: two demonyms and three nouns which are not proper nouns, all with possessive and plural forms (bar one which is its own plural).*

grep -i '^p' words |

* S/O - Standard Output
S/I - Standard Input

A pipe is the symbol on the keyboard that is a tall vertical line. It is used to redirect the output of one command to the input of another (rather than being sent to standard output)

Let's look at an example:

`ls | head`

The commands on their own:
ls - lists content of current directory
head - displays the first few lines of its input (default is 10 but has optional head -n flag)

So together this command displays the first 10 items in a directory.

Other useful commands relating to pipes:

cat [FILENAME [FILENAME ...]]
This writes the content of one or more files to S/O. This is a good way of starting pipes. If you leave the filename section blank, cat just reads its *S/I & writes to S/O

head [-n N]
This reads its S/I & writes only the first N lines. If N is unspecified the default is 10
N can also be less than zero. This will skip the last N lines
e.g in a 10 line script prog.c
cat prog.c | head -n 8 ≡ cat prog.c head -n -2

tail [-n N]
Similar idea to head except this prints the last N lines & if N<0 it skips the first N lines

sort
This reads the contents of its S/I into a memory buffer, sorts the lines & writes them all to S/O

uniq
This reads its S/I & writes to S/O but it skips repeated lines that immediately follow eachother i.e
uniq [A A B] → [A B]
uniq [A B A] → [A B A]
A common way to remove duplicates is to do:
cat [FILENAME] | sort | uniq

grep [-iv] EXPRESSION
This reads its S/I & returns only the lines that match the regular expression. -i creates case-sensitivity & -v inverts it (only prints lines that don't match). Words only need to contain these expressions
i.e grep "p" returns all lines containing a 'p'
PTO

There are 3 other key symbols used here:

^ (hat), this means "The start of the line"
e.g grep -i "^s" will return all lines that start with a capital S.

$ this is "The end of the line"
grep "s$" will return all lines that end in s/S.

. (fullstop) means any character so:
grep "^....$" will return all lines 4 characters long.

sed -e COMMAND

This reads from its S/I & transforms it according to the given command & writes the result to the S/O. sed stands for stream editor & has its own command language. The most common is:
's/SOURCE/DEST/g'

Example:
`sed -e 's/p/q/g' prog.c`
This will set all lower case p's to lower case q's in prog.c

wc [-l]

wc stands for word count. This is most often used with the -l flag, this returns the number of lines (entries) returned. With no flag, wc returns the number of lines, the number of words (-w) & the number of bytes (-c)

If you're unsure on how to use any of these commands you can use the man command
This is short for manual usage

`man [INSTRUCTION]`