Running external programs

back-ticks, **`system()`**, and pipes

OpSys – like functions

- Those of you who have taken CSCI-4210, Operating Systems, should be aware that Perl has the OpSys functions you would expect:
  - **`fork()`**, **`exec()`**, **`waitpid()`**, etc
- We will not be discussing these functions
  - I don't want to be responsible for the results
  - feel free to examine **`perldoc perfunc`** for the full listing
  - We may cover them during the "Advanced Perl Topics" lecture towards the end of the semester.

`system(CMD, LIST)` `system(CMD_and_ARGS)`

- Takes a pathname, followed by a list of arguments.
- Executes the program specified by the pathname
  - To use shell meta-characters, give only one argument – a string containing the path and arguments.
- **`system ('myprog.pl', 'input.txt');`**
- **`system ('myprog.pl ~john/input.txt');`**
- Returns the **return value** of the command that was executed….. and some more
  - You have to shift the result right 8 bits
  - **`my $ret = system( … );`**
  - **`$ret = $ret >> 8;`**

backticks

- Third kind of quoting operator
- Enclose a pathname.
- Executes the command specified by the pathname
- Returns the **output** of that command.
  - single string in scalar context
  - one line per element in list context
- `my @files = `ls –al`;`
- @files contains one file listing per element

more on backticks

- Just like double quotes, backticks are subject to interpolation.
  - `$cmd = 'prog1.pl';`
  - `$output = `$cmd –h`;`
- like `q//` for single quotes, and `qq//` for double quotes, backticks can be represented by `qx//`
  - same delimiter rules apply

Pipes

- the **open** function can link a filehandle to a running process, instead of a file.
- To write to a process, specify mode of `'|-'`
- To read from a process, specify mode of `'-|'`

```
open my $mh, '|-', '/bin/sendmail' or die(…);
open my $ls, '-|', 'ls –al' or die(…);
```

- Alternatively (but not recommended), 2-arg version:
  - `open my $mh, '| /bin/sendmail' or die(…);`
  - `open my $ls, 'ls –al |' or die (…);`
- Once pipes have been opened, read/write them just as any other filehandle.
  - `while (<$ls>) { print $mh "File: $_"; }`

Piping issues

- Why pipe instead of using ``?
  - with pipes, you can read one line of output at a time, and terminate process at any time, using **`close()`**
- Opening a process for bi-directional communication is more complicated.
  - Involves using the IPC modules.
  - See Camel page 430 for more info.
    - **`perldoc –f open`**
    - **`perldoc perlopentut`**
    - **`perldoc IPC::Open2`**