Ruby 101

Advanced Ruby: Sub-Processes Etc

Advanced Ruby:

Executing Outside Programs:

`*command*` -> using backticks (top left corner of the keyboard
returns the output from standard output
status recorded in $?

ex

my_user_name = `whoami`

Advanced Ruby:

Executing Outside Programs:

`exec 'command'` -> replaces current process with *command*

ex

```
puts "Ruby Process"
exec 'whoami'
puts "This will never get printed out"
```

Ruby Process
username

Advanced Ruby:

Executing Outside Programs:

```
system 'command'
```
-> runs *command* in a subshell, waits for command to complete

ex

```
puts "Ruby Process"
system 'whoami'
puts "This will get printed out"
```

Ruby Process
username
This will get printed out

Advanced Ruby:

Executing Outside Programs:

```
pid = fork do
    ...
end
```

=> Creates a child process, by forking the current one. Only the calling thread will be accessible to child returns twice (PID in parent; nil in child)

```
Process.wait [pid]
```

=> waits for child processes to finish, allows them to exit; optional pid can be specified to wait for that process

```
waiting_thread = Process.detach pid
```

=> Inform operating system that child processes are no longer watched... allow them to exit

Advanced Ruby:

Executing Outside Programs:

```
io = IO.popen command, ['mode']
IO.popen command, ['mode'] do |io|
    ...
end
```

=> Creates IO object connecting a program's standard i/o to the io object

.close => closes io object