Multiprocess Parallelism

- parallelism — running things simultaneously
- concurrency — running things independently

Lorikeet (extension of Falcon)

<expr> ::= ..... | sleep(<expr>) — waits for some number of seconds

| parallel(<expr>) — takes a closure expecting one argument
  - run function seperately
  - pass in a "channel"
  - return a matching channel

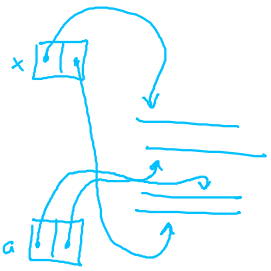
| send(<expr>, <expr>) — transmit second value on the first value (which must be a channel)

| receive(<expr>) — read a value from the provided channel (will block until a value is available)

```
def f x =
    let _ = sleep(1) in
    print(5)
end
let _ = parallel(f) in
print(4)
```

5 4 4
4 5 4
4 4 5

```
def f x =
    let n = receive(x) in
    print(n)
end
let a = parallel(f) in
let _ = print(2) in
let _ = send(a, 4) in
6
```



2 2
4 6
6 4

Implementation

every time parallel is invoked:

- create two pipes to use as channel
- fork process
- parent process: evaluate to channel
- child process:
  - create channel
  - pass channel to function
  - exit

pipe — OS-managed queue

identified by a "file descriptor"

"channel" — bird value in the form

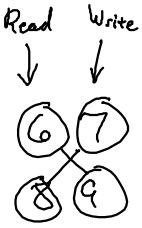
0x RRRRRRR WWWWWWW 03

(7 bits) (7 bits)

RRRRRRR: file descriptor read queue

WWWWWWW: file descriptor write queue

03: 011



0x0000006000000903

0x0000008000000703

send only ints and bools in bird form

Fork

	virtual memory		physical memory	copy-on-write
→	0x7ffffff700030	cow	0x57000	
→	0x7ffffff700030		disk@	