UNIX-FU:

A FORKING PRESENTATION FOR DEVELOPERS

https://github.com/gregmalcolm/unix_for_programmers_demo
https://gist.github.com/1241642
@gregmalcolm

TIMELINE

1969 - UNIX is Born

1972 - Rewritten in C

1977 - BSD UNIX

1981 - IBM PC

1983 - Richard Stallman founds FSF

1991 - Linus Tovalds starts Linux

TIMELINE

PHILOSOPHY

Doug McIlroy (inventer of unix pipeline):

"This is the Unix philosophy:

Write programs that do one thing and do it well.
Write programs to work together.
Write programs to handle text streams,
because that is a universal interface."

PHILOSOPHY

Doug McIlroy (inventer of unix pipeline):

"This is the Unix philosophy:

Write programs that do one thing and do it well.
Write programs to work together.
Write programs to handle text streams,
because that is a universal interface."

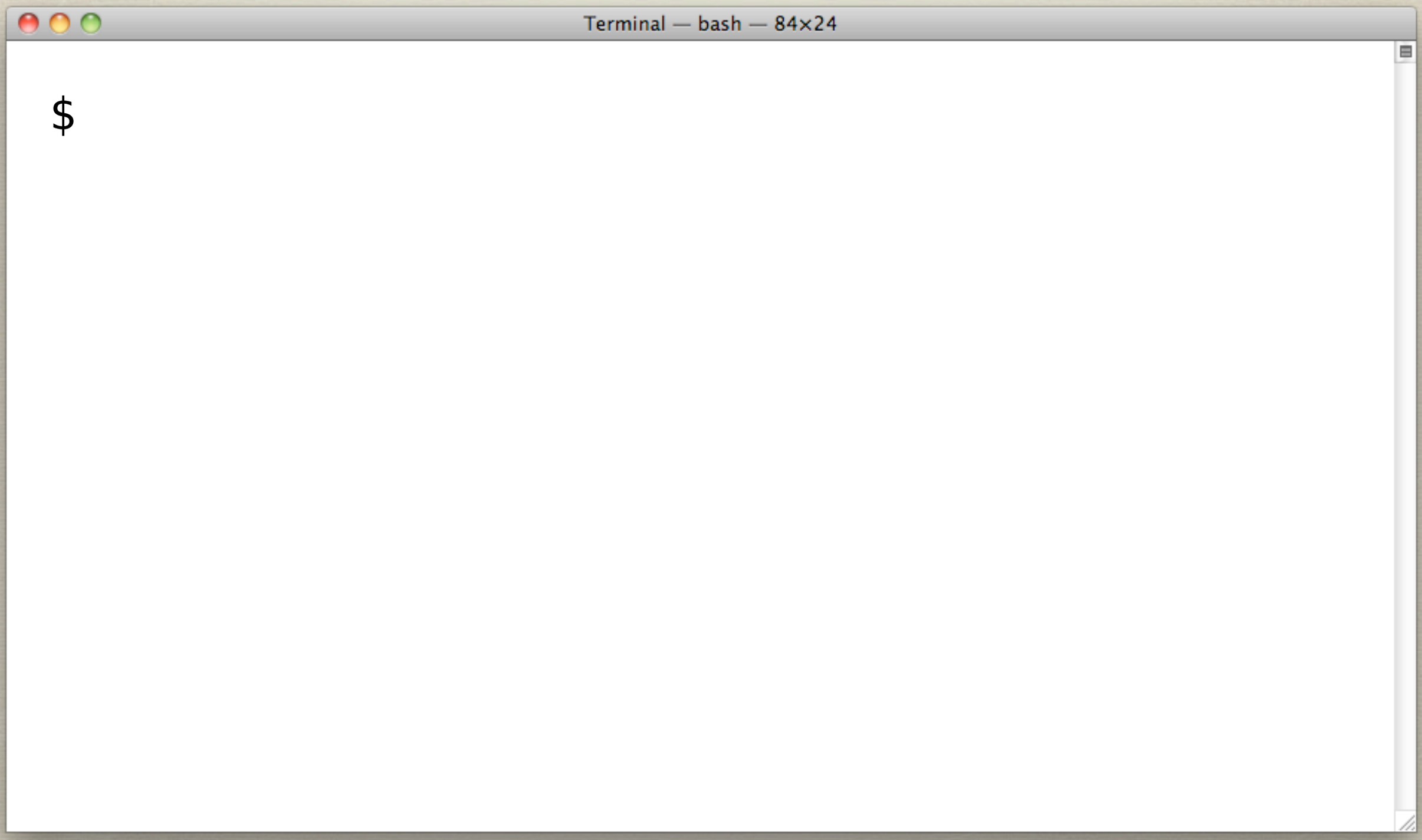
PHILOSOPHY

Mike Gancarz (part of X Windows team):

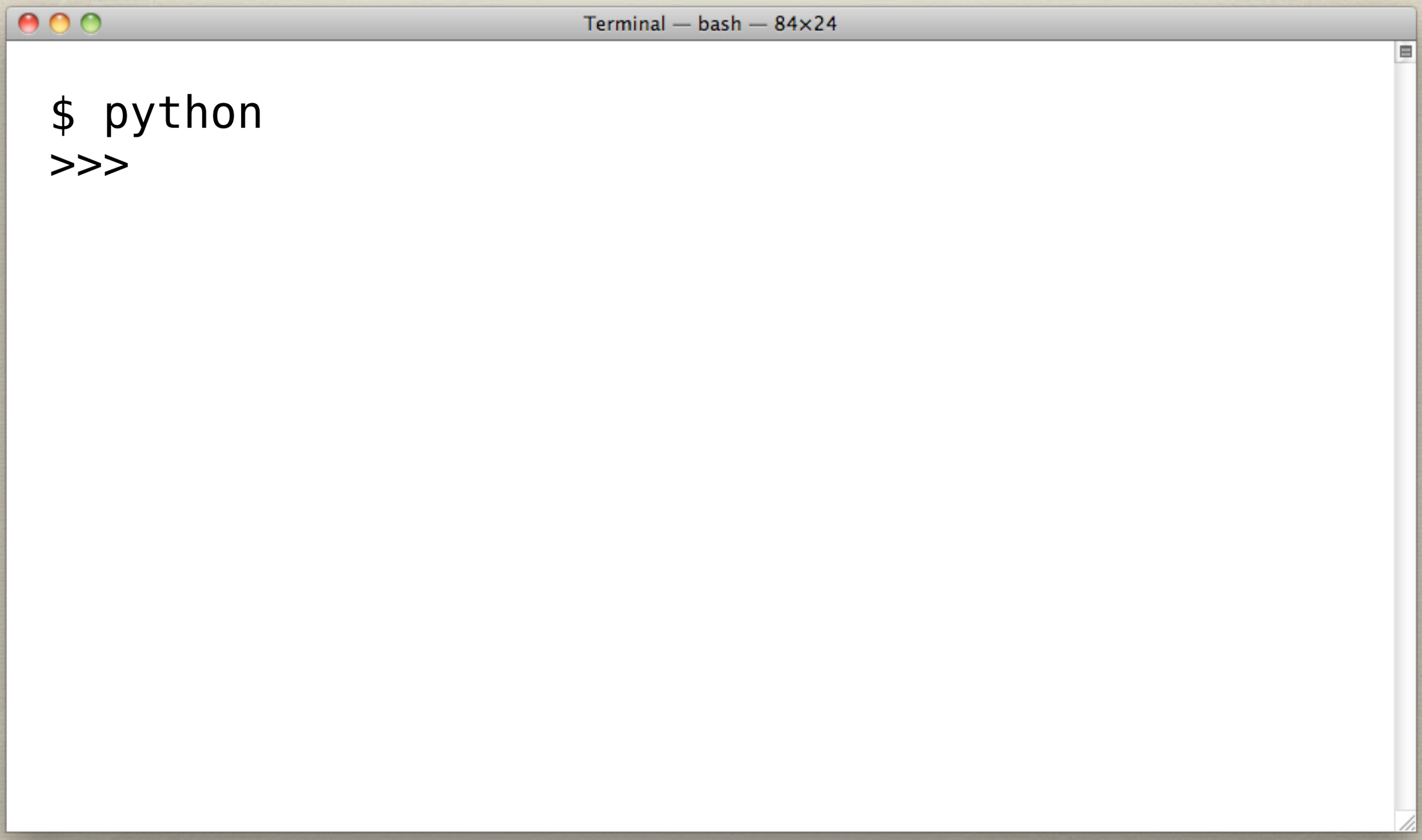
" *1. Small is beautiful.*
2. Make each program do one thing well.
3. Build a prototype as soon as possible.
4. Choose portability over efficiency.
5. Store data in flat text files.
6. Use software leverage to your advantage.
7. Use shell scripts to increase leverage and portability.
8. Avoid captive user interfaces.
9. Make every program a filter."


Don't cross the streams!
ICANHASCHEEZBURGER.COM

STREAMS
File Streams



STREAMS
File Streams



STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>>
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
>>>
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
>>> f2 = open('file2.in', 'r')
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
>>> f2 = open('file2.in', 'r')
>>>
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
>>> f2 = open('file2.in', 'r')
>>> f.fileno()
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
>>> f2 = open('file2.in', 'r')
>>> f.fileno()
3
>>>
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
>>> f2 = open('file2.in', 'r')
>>> f.fileno()
3
>>> f2.fileno()
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
>>> f2 = open('file2.in', 'r')
>>> f.fileno()
3
>>> f2.fileno()
4
>>
```

STREAMS
File Streams

Terminal — bash — 84×24

```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
>>> f2 = open('file2.in', 'r')
>>> f.fileno()
3
>>> f2.fileno()
4
>>> f2.read()
```

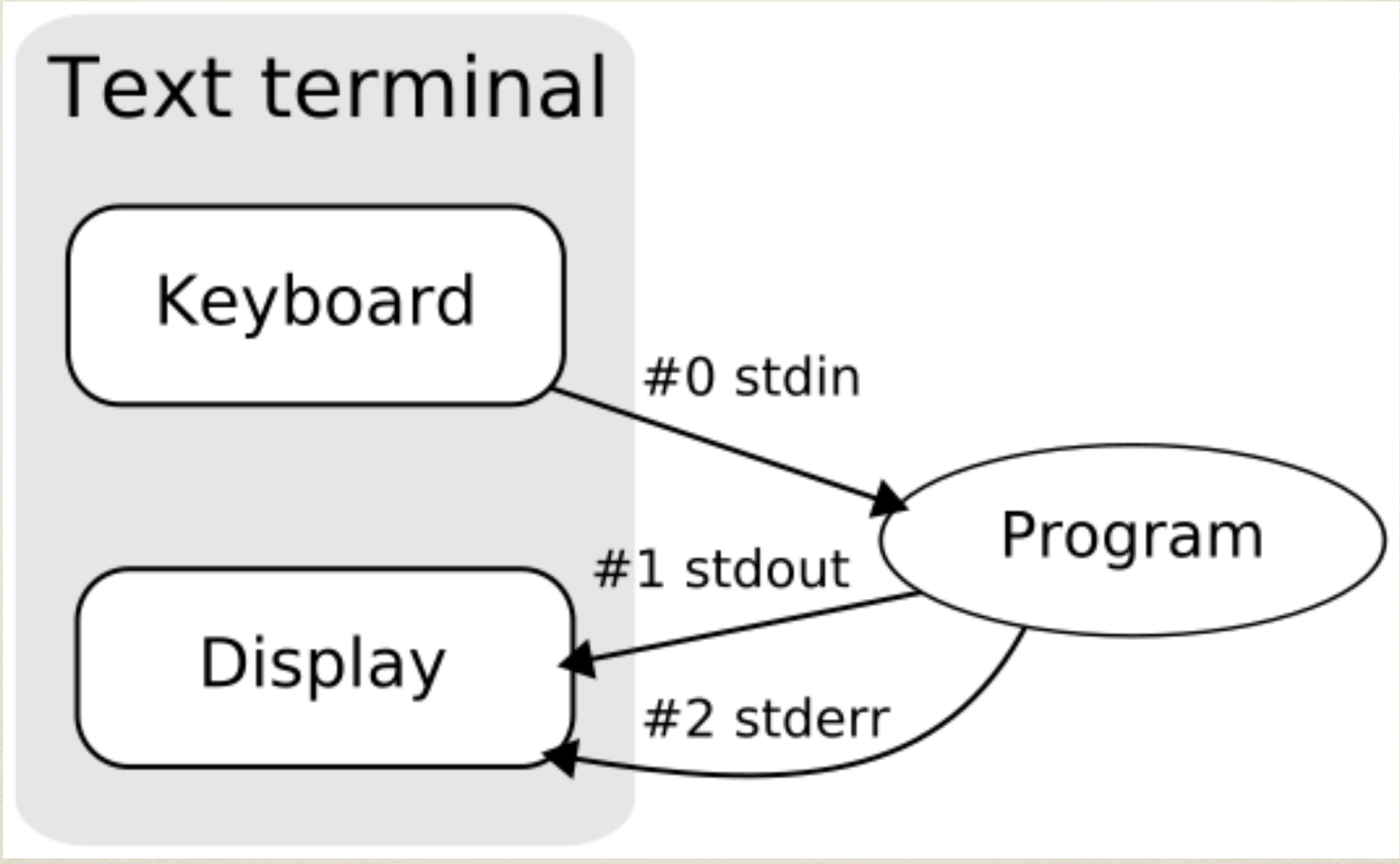
STREAMS
File Streams

Terminal — bash — 84×24

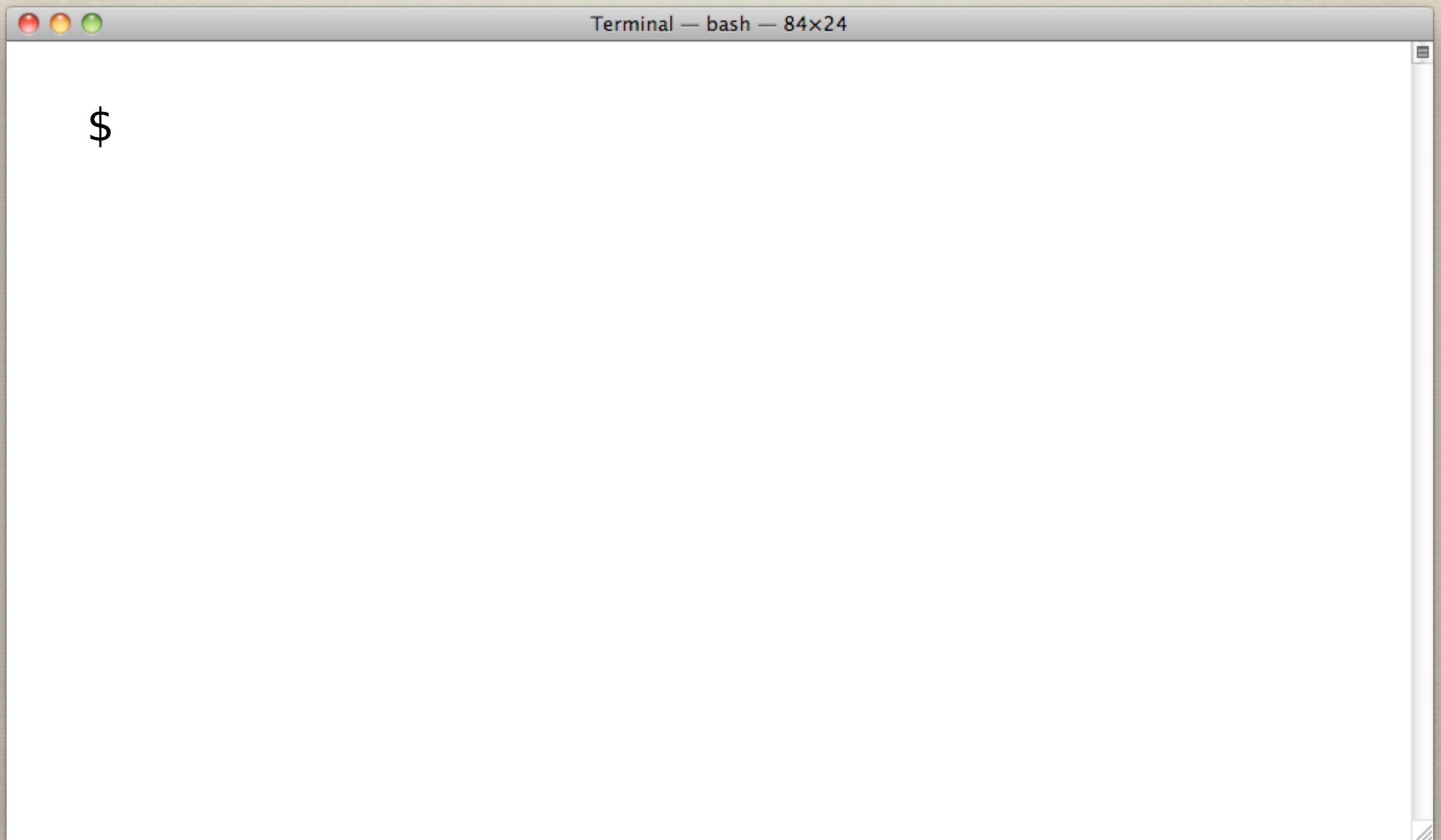
```
$ python
>>> f = open('file1.out', 'w')
>>> f.write("Some text")
>>> f2 = open('file2.in', 'r')
>>> f.fileno()
3
>>> f2.fileno()
4
>>> f2.read()
'Sample input\n'
```

STREAMS

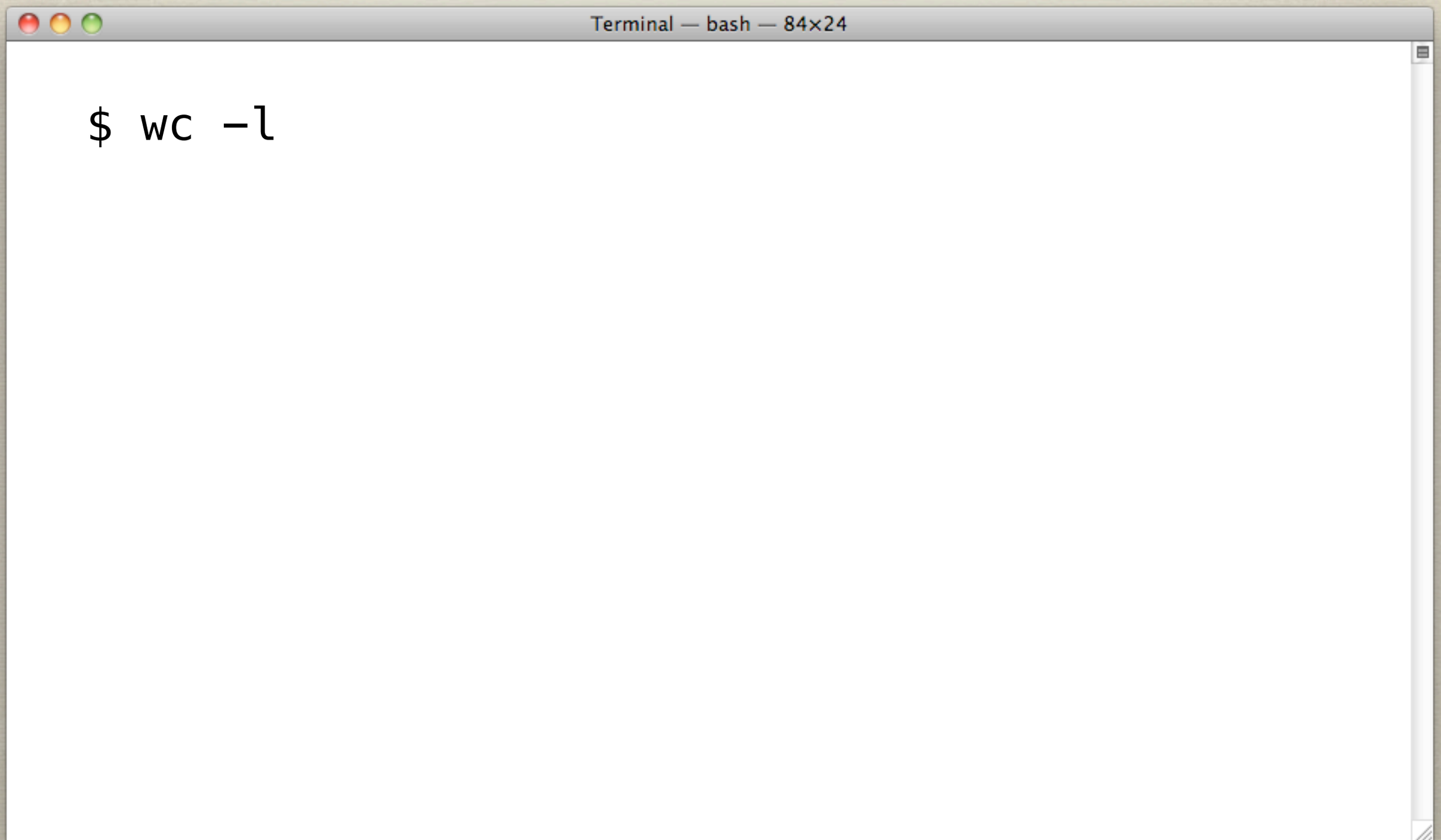
Input/Output Streams



STREAMS
STDIN

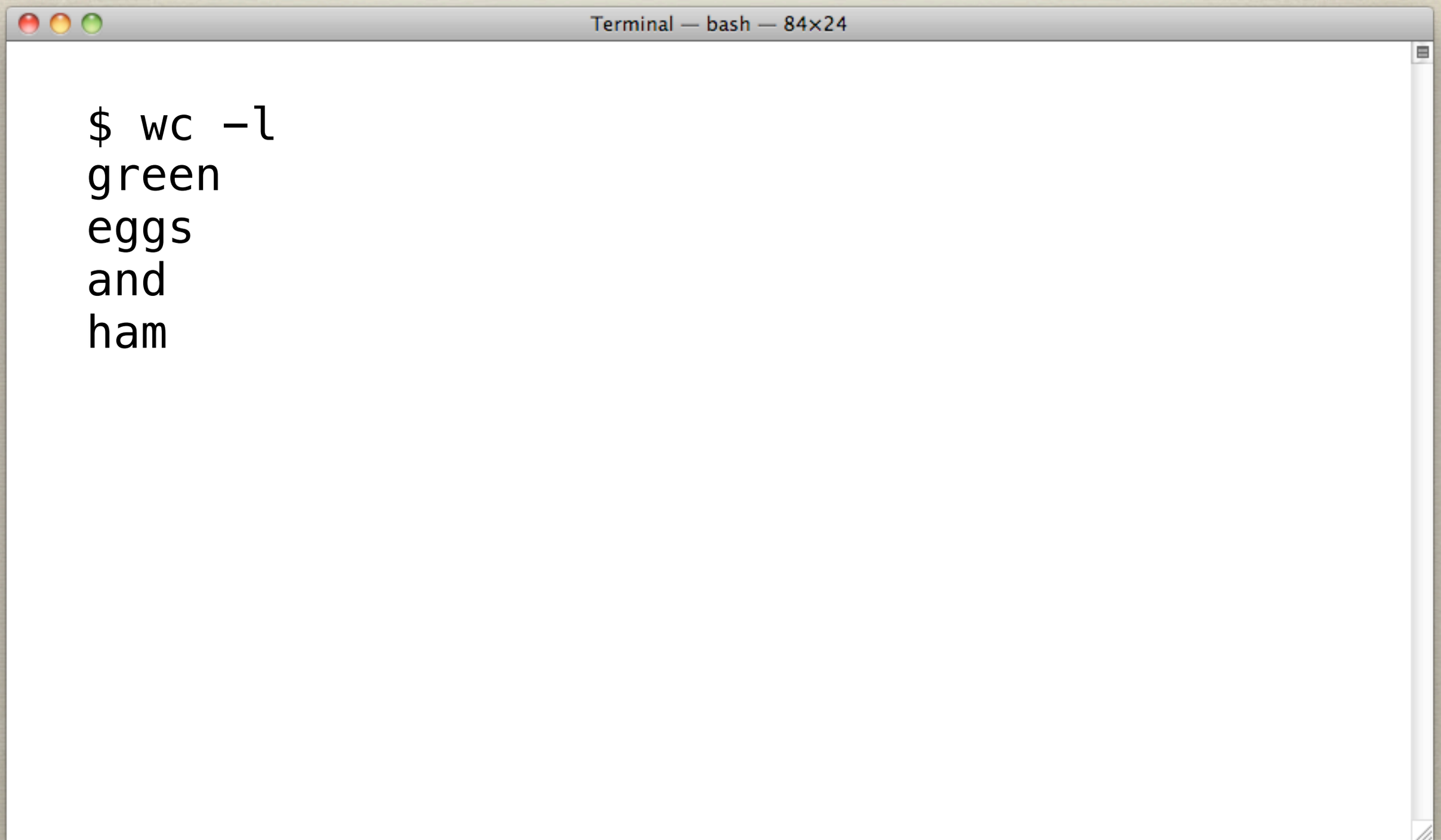


STREAMS
STDIN



STREAMS

STDIN



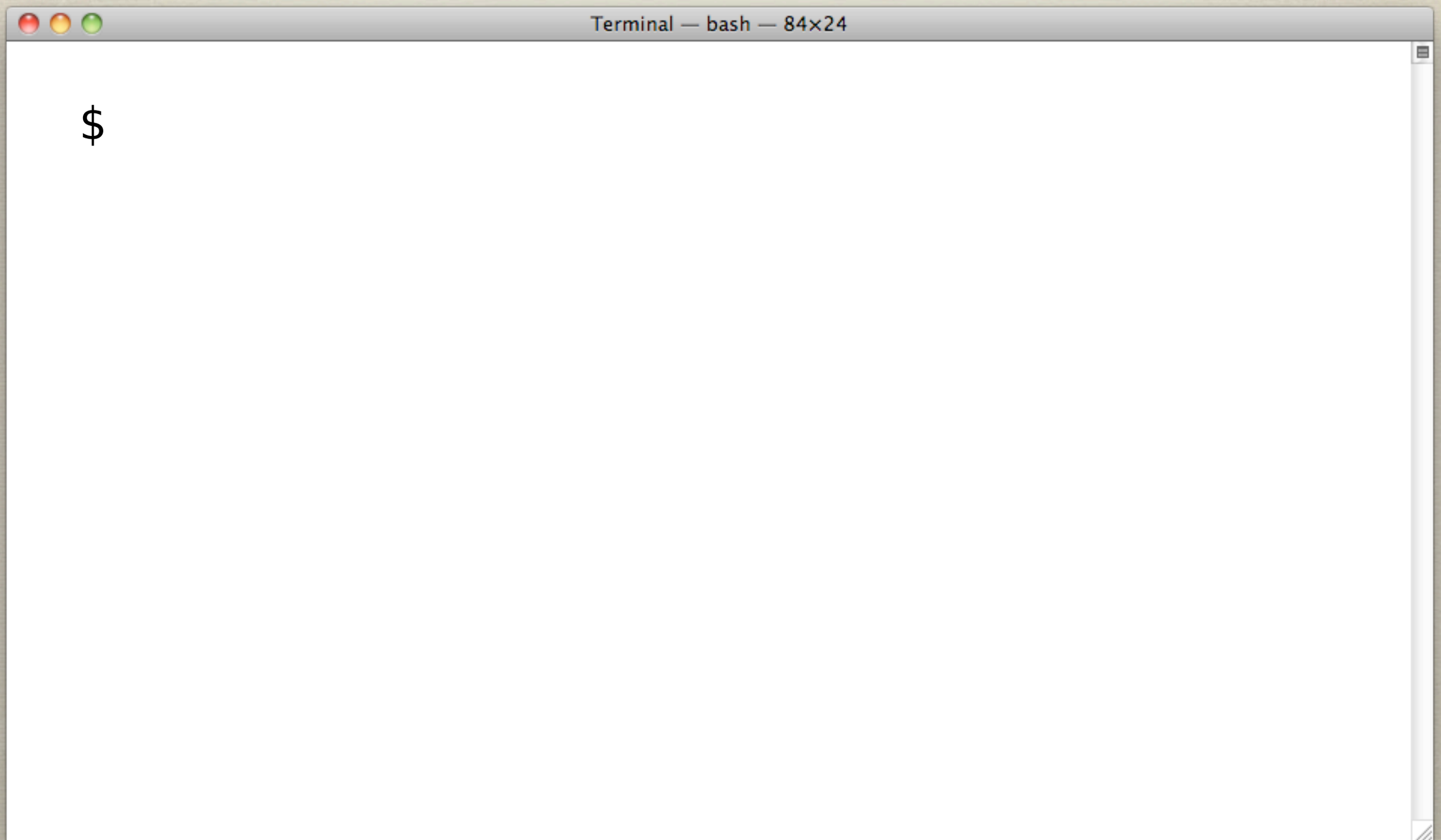
STREAMS

STDIN

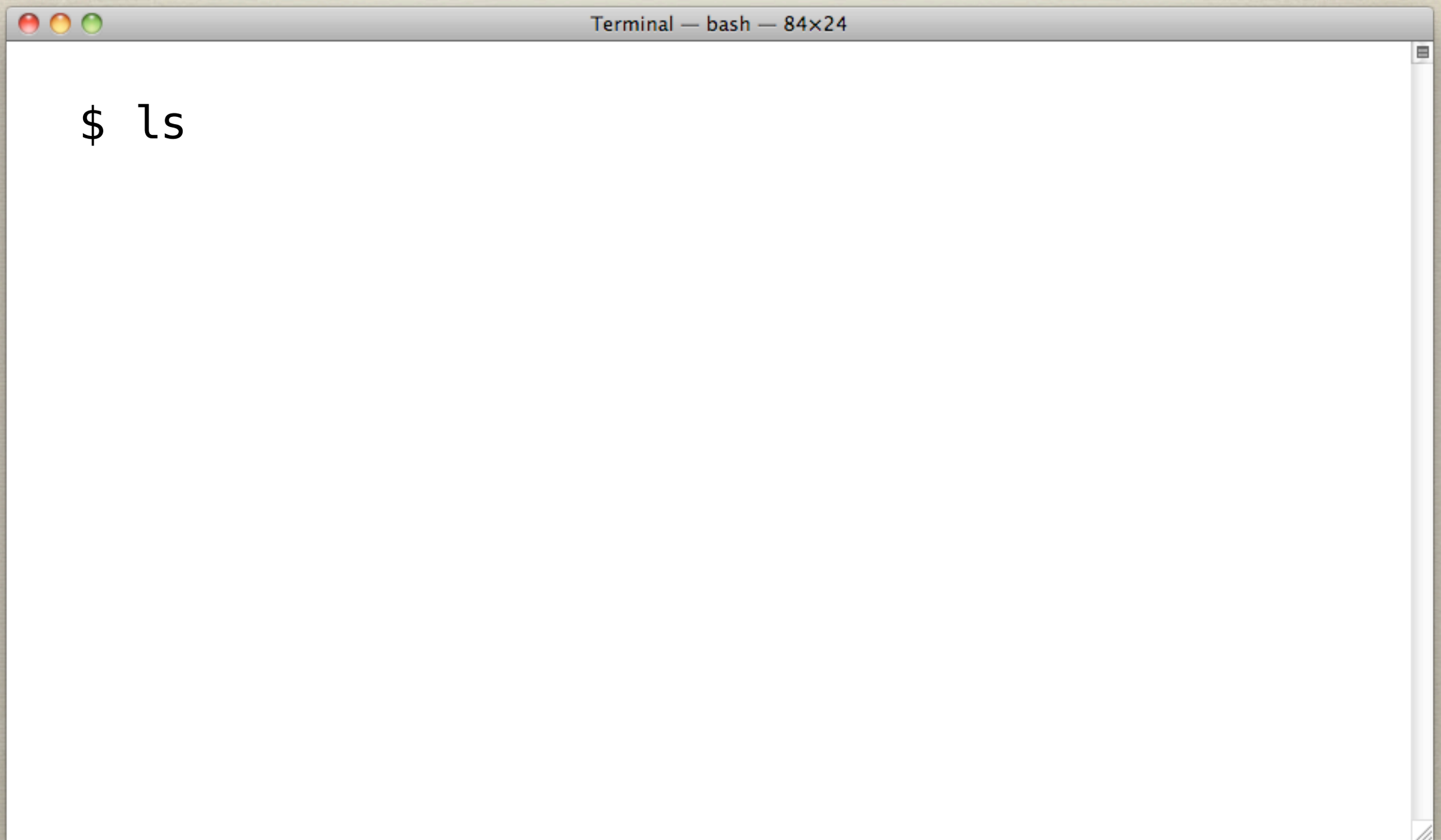
Terminal — bash — 84×24

```
$ wc -l
green
eggs
and
ham
       4
$
```

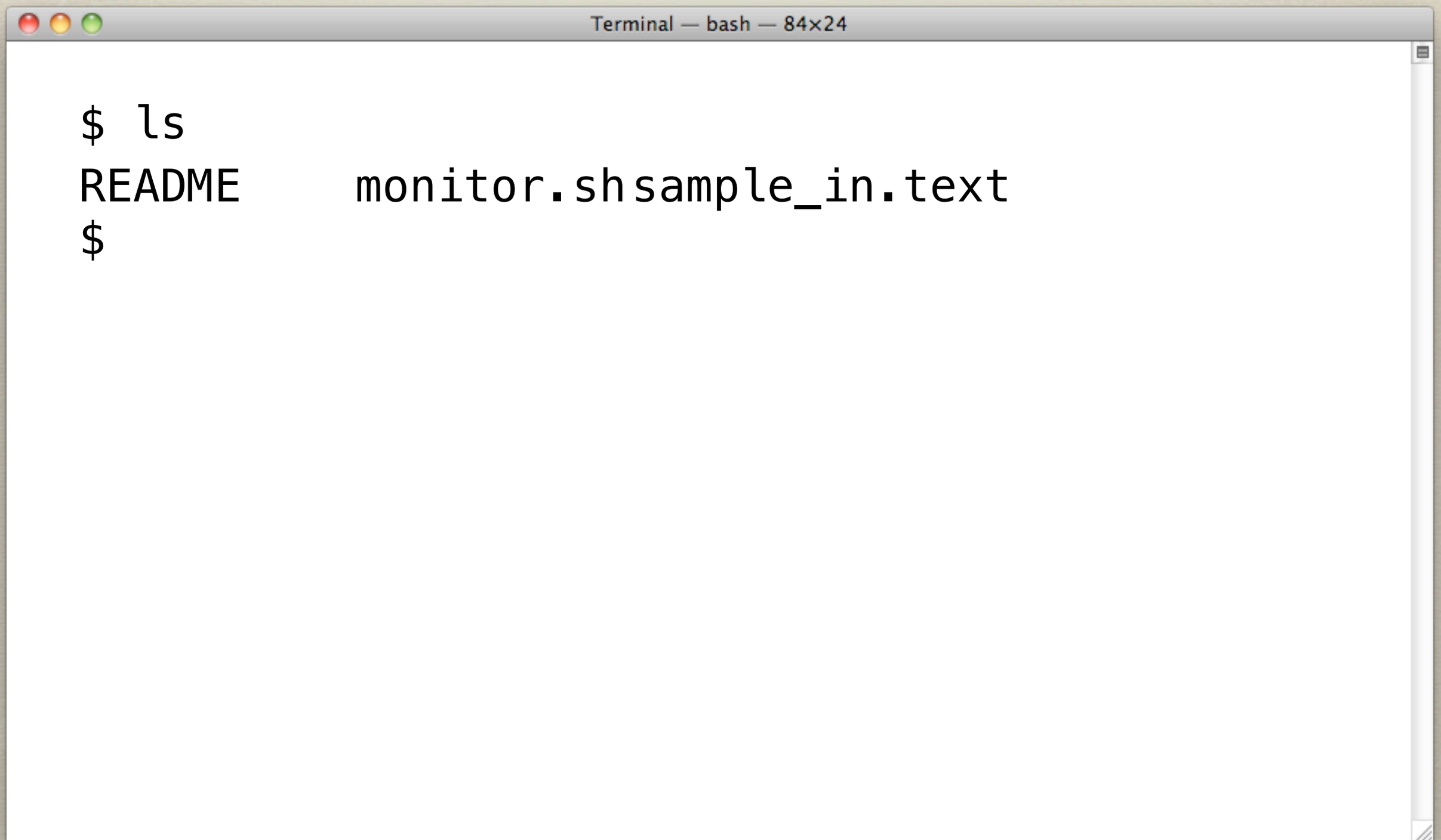
STREAMS
STDOUT



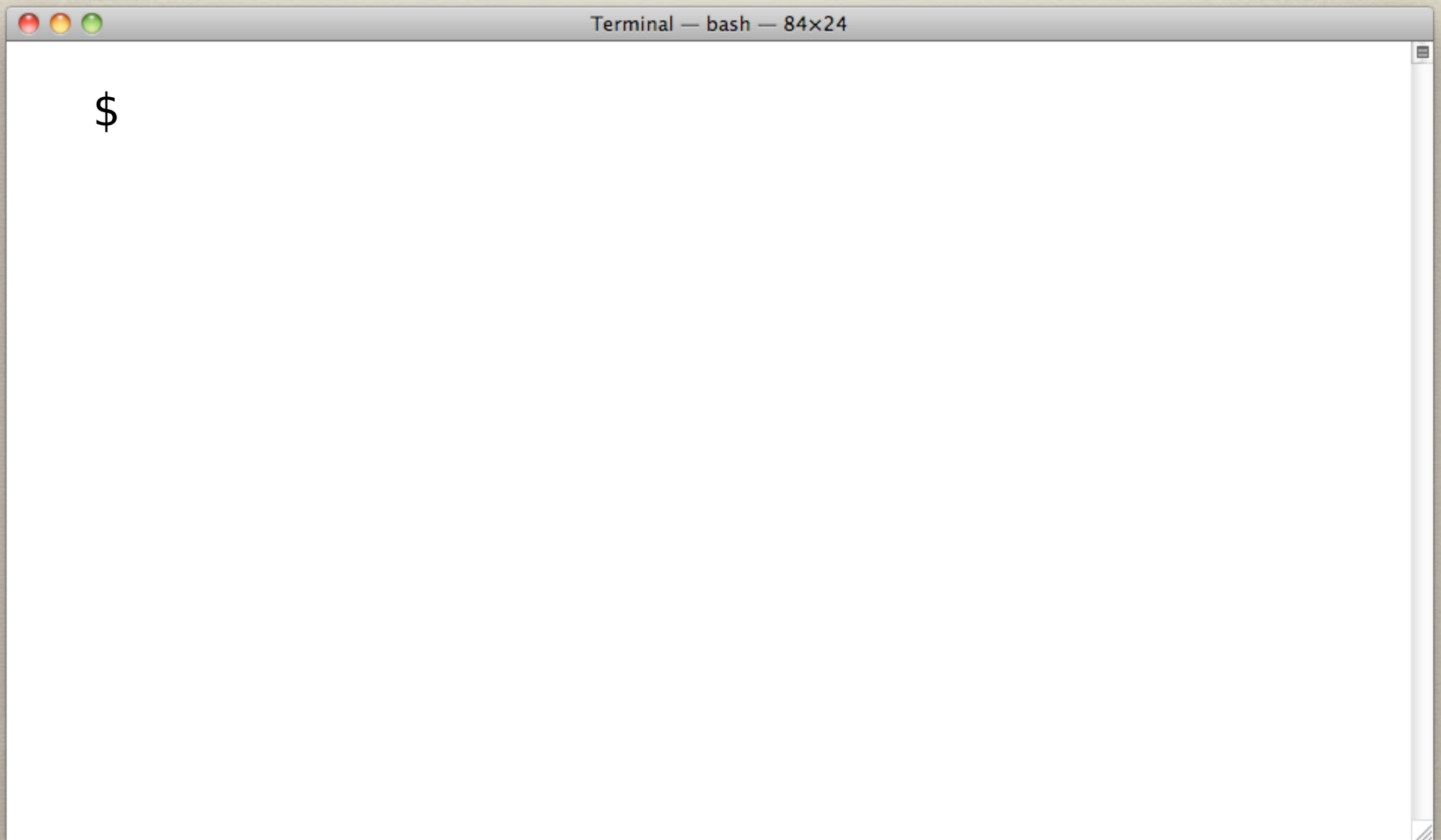
STREAMS
STDOUT



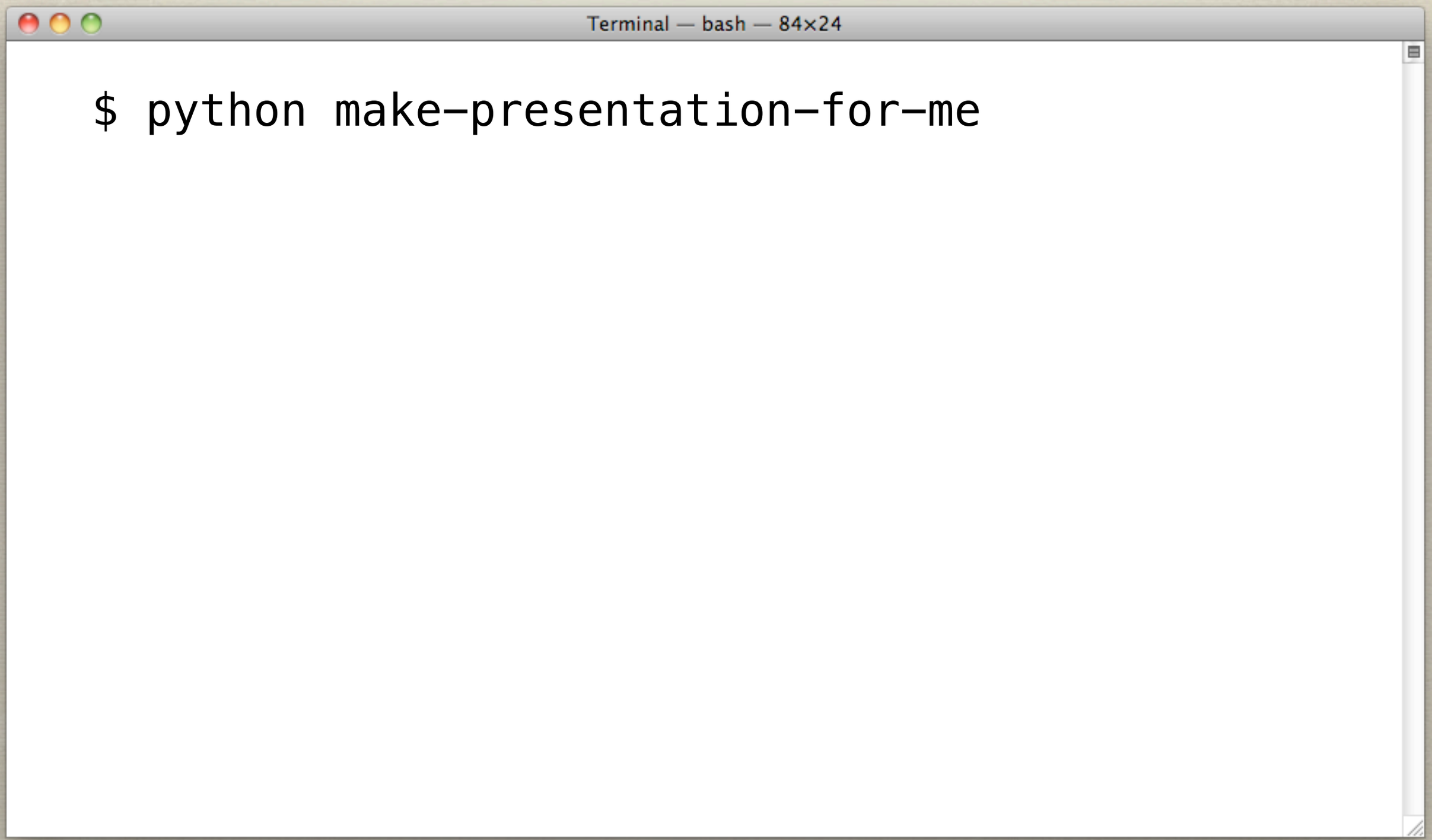
STREAMS
STDOUT



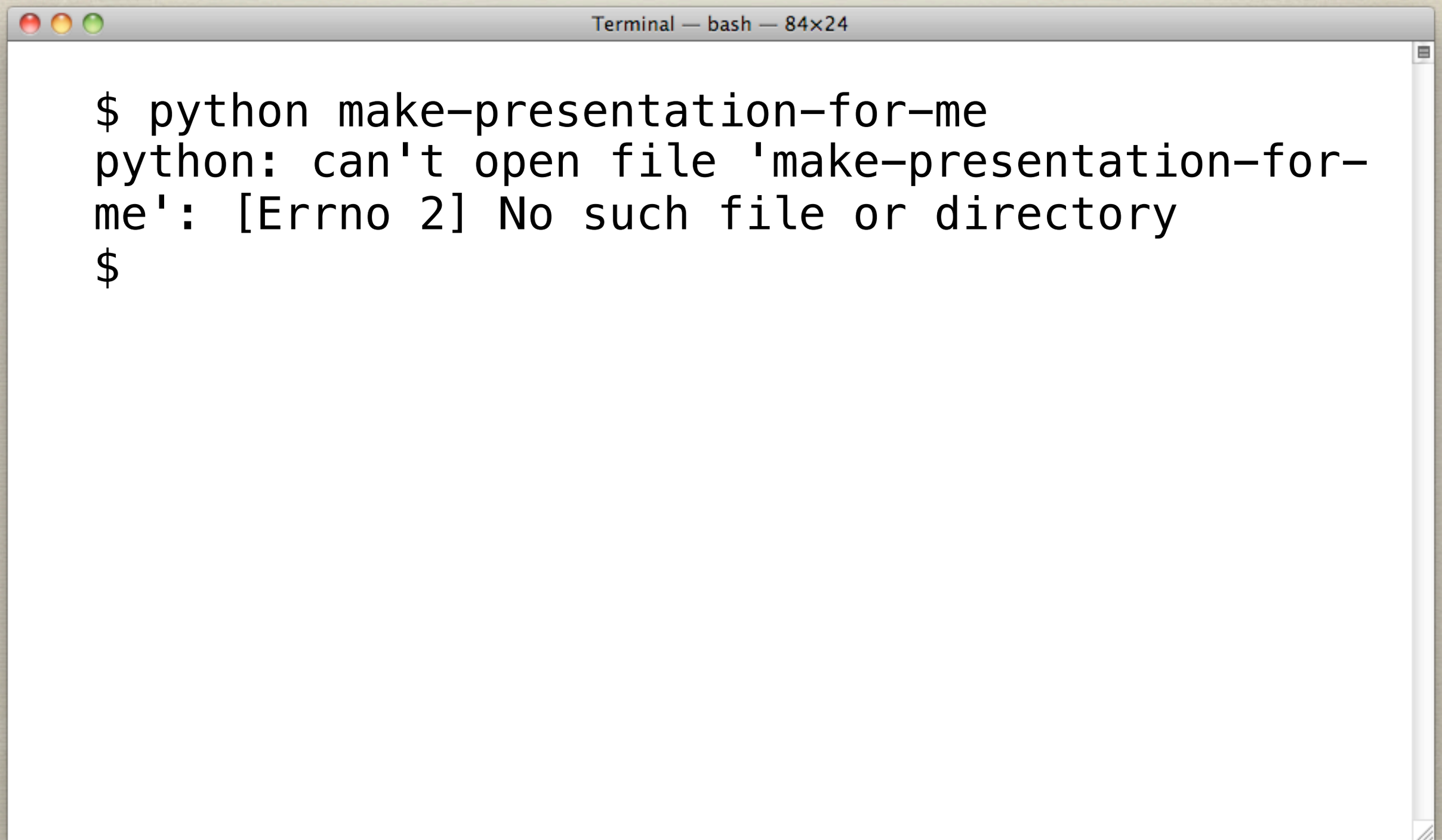
STREAMS
STDERR



STREAMS
STDERR

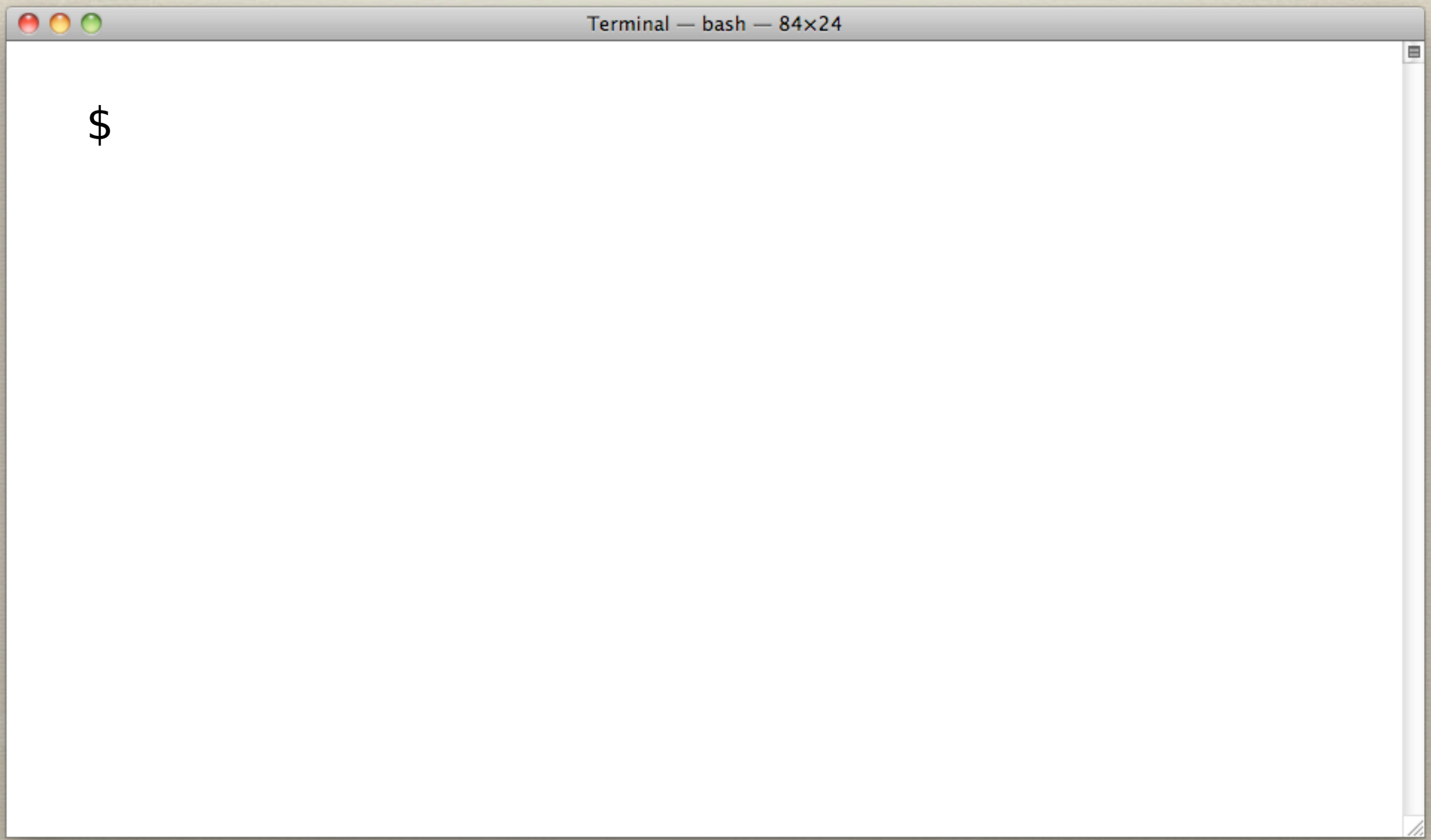


STREAMS
STDERR



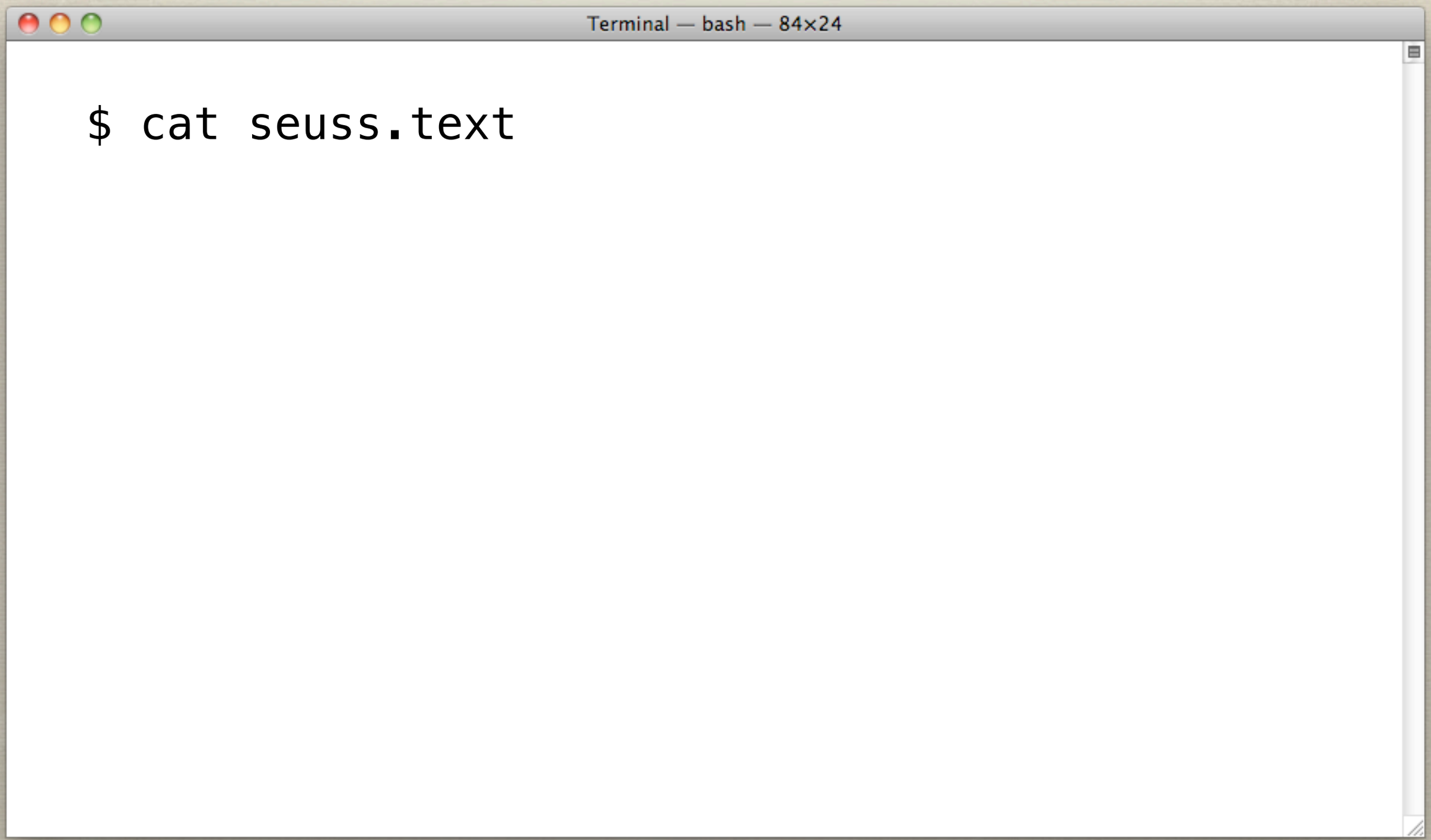
STREAMS

Redirecting STDIN



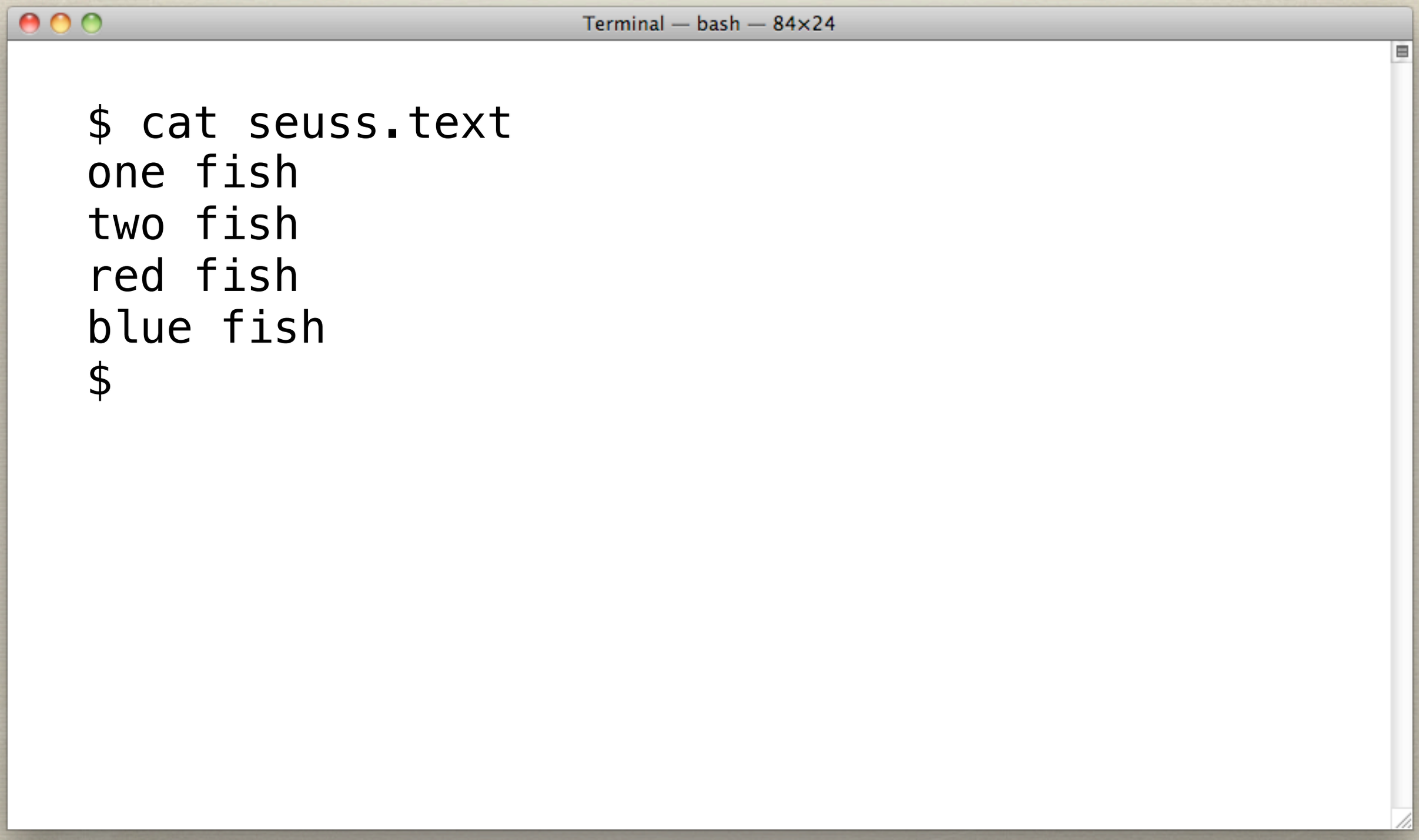
STREAMS

Redirecting STDIN



STREAMS

Redirecting STDIN



STREAMS

Redirecting STDIN

Terminal — bash — 84×24

```
$ cat seuss.text
one fish
two fish
red fish
blue fish
$ wc -w <seuss.text
```

STREAMS

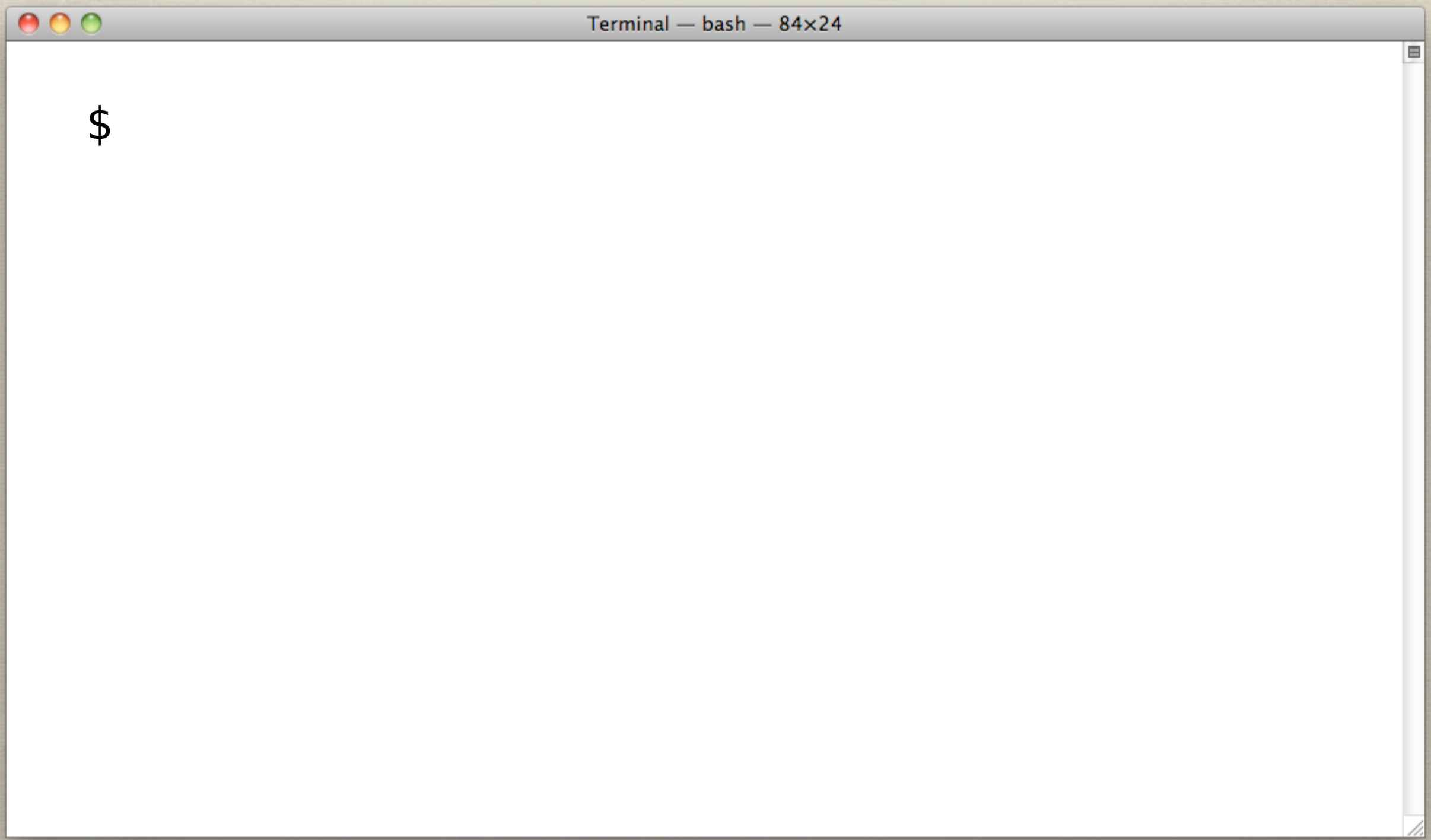
Redirecting STDIN

Terminal — bash — 84×24

```
$ cat seuss.text
one fish
two fish
red fish
blue fish
$ wc -w <seuss.text
       8
$
```

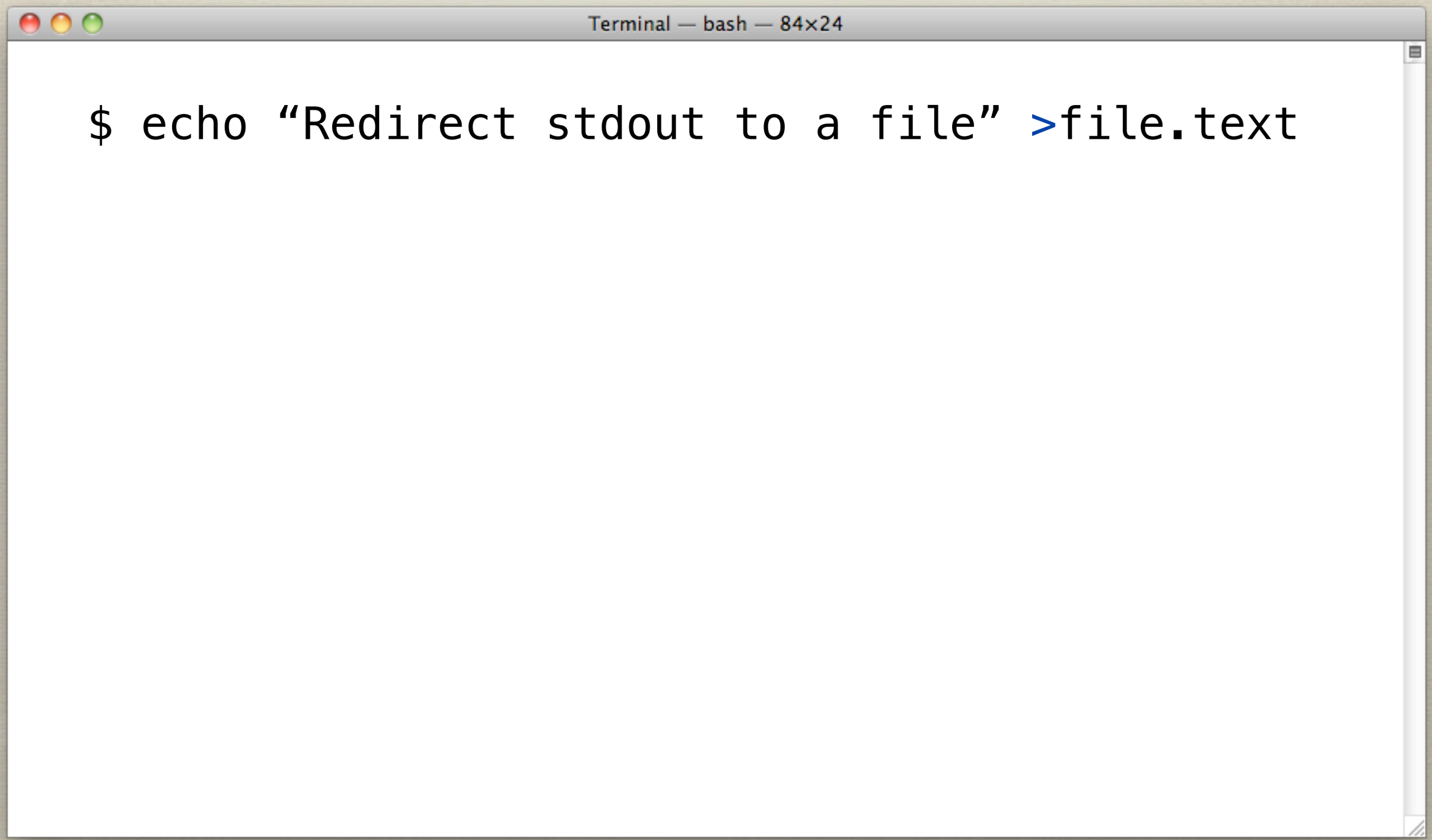
STREAMS

Redirecting STDOUT and STDERR



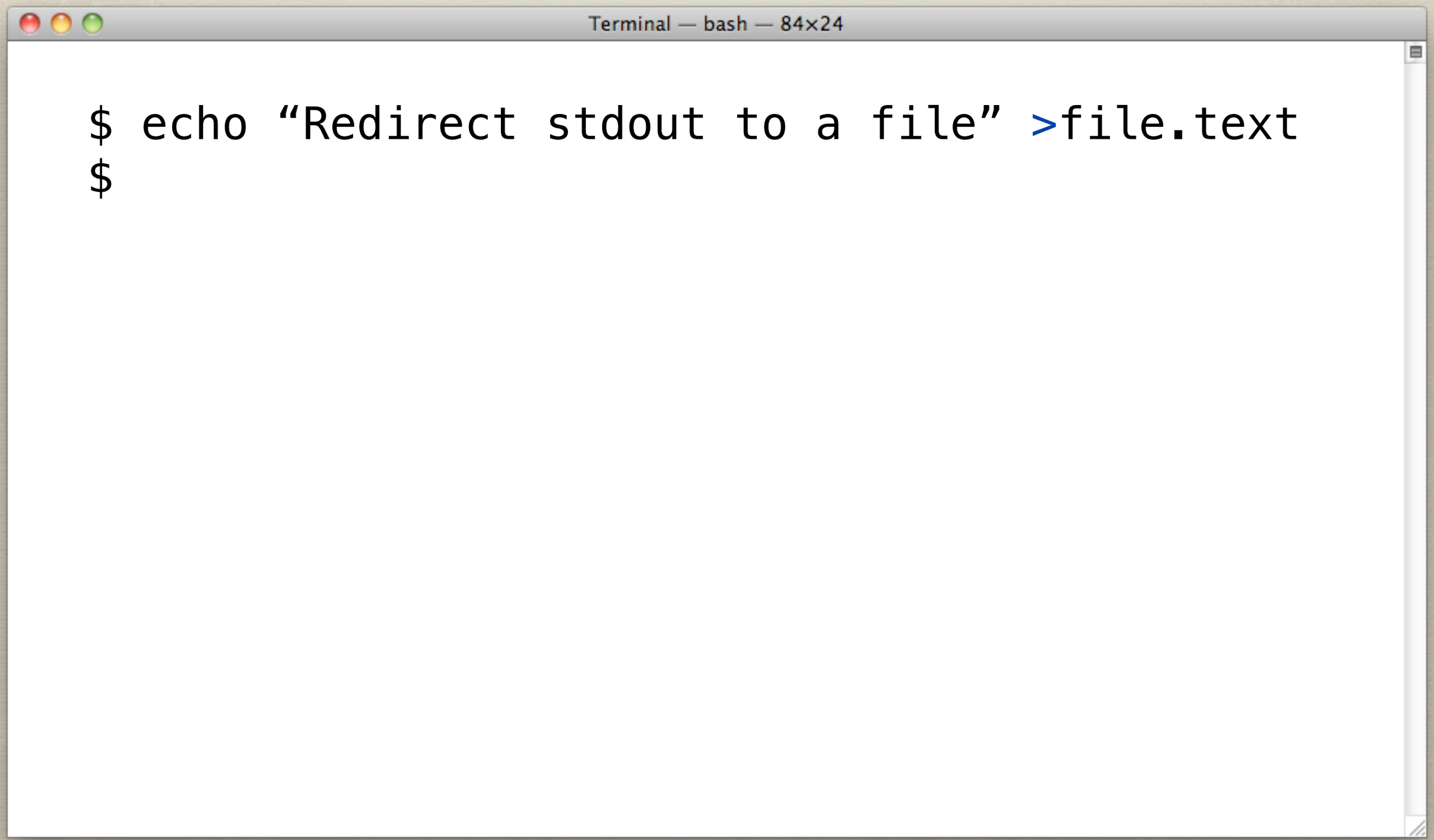
STREAMS

Redirecting STDOUT and STDERR



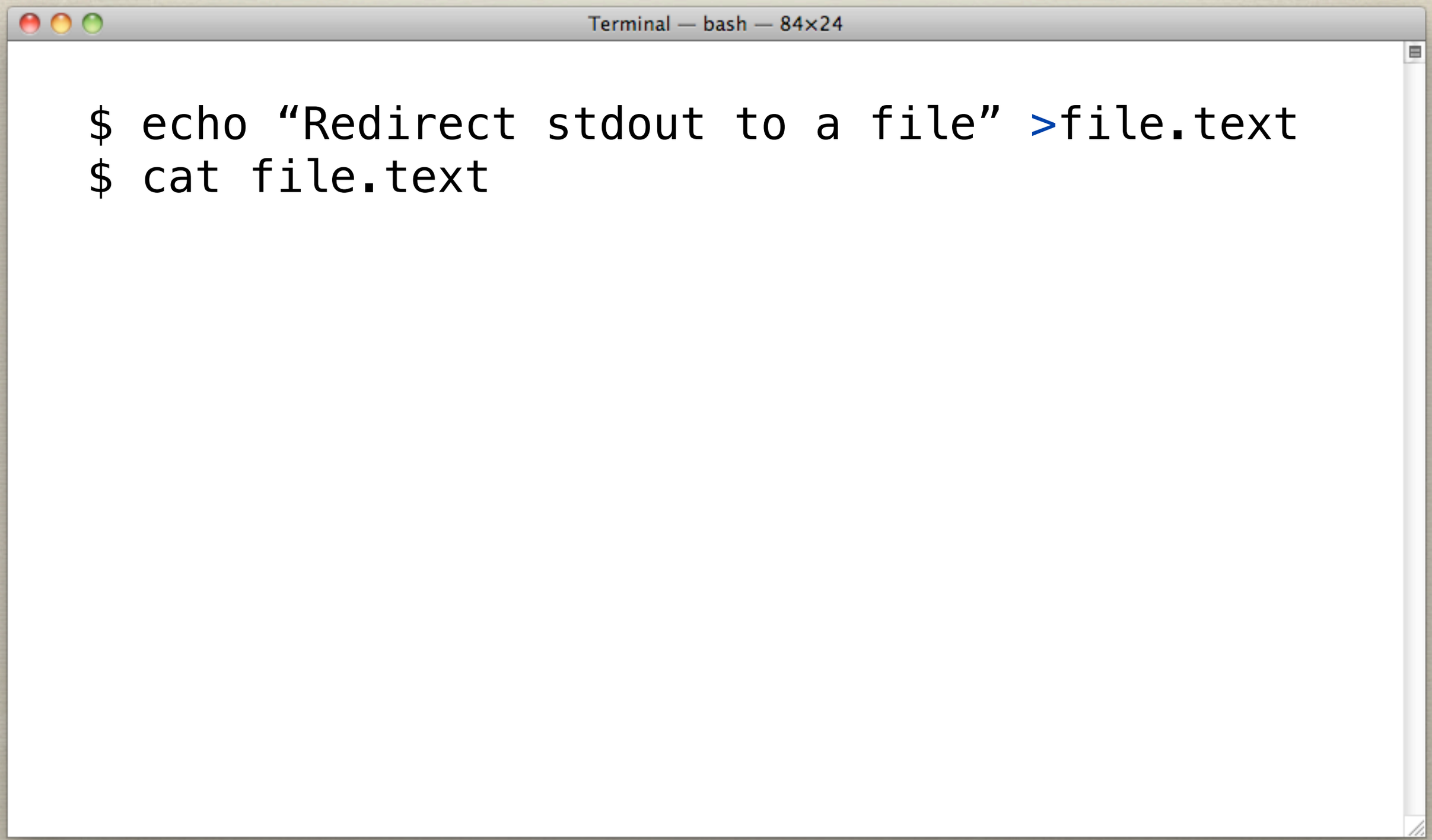
STREAMS

Redirecting STDOUT and STDERR



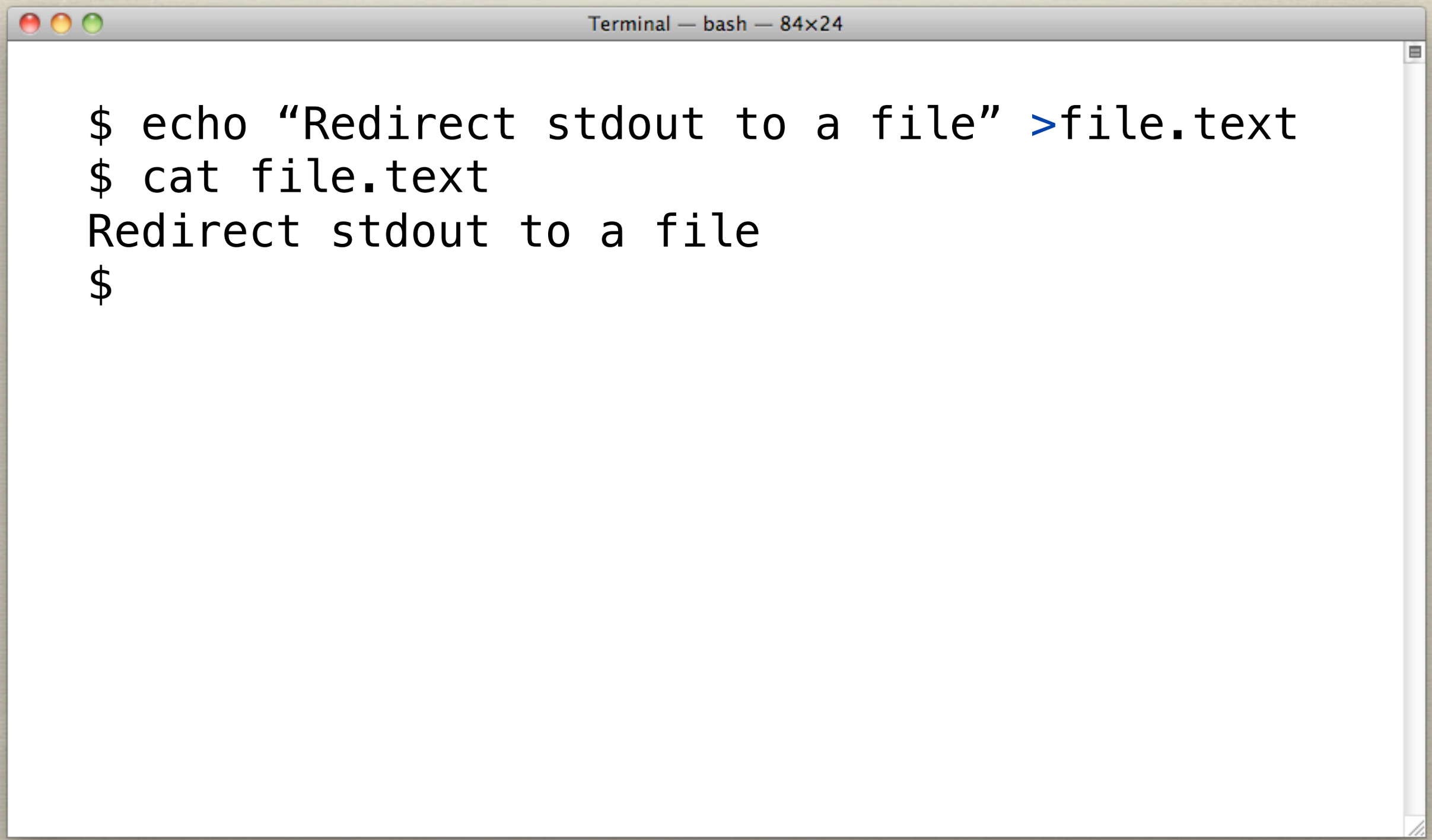
STREAMS

Redirecting STDOUT and STDERR



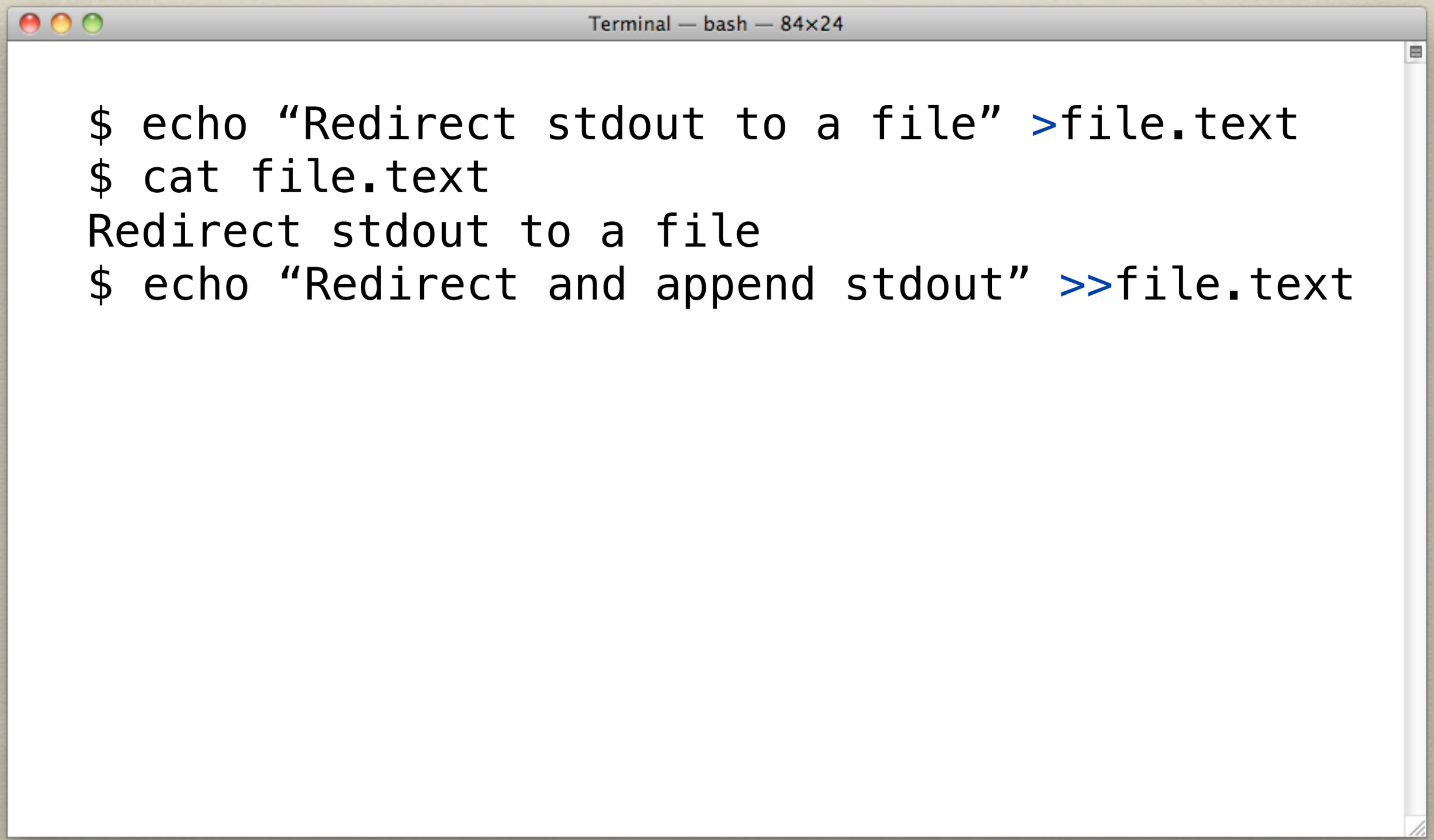
STREAMS

Redirecting STDOUT and STDERR



STREAMS

Redirecting STDOUT and STDERR



STREAMS

Redirecting STDOUT and STDERR

Terminal — bash — 84×24

```
$ echo “Redirect stdout to a file” >file.text
$ cat file.text
Redirect stdout to a file
$ echo “Redirect and append stdout” >>file.text
$
```

STREAMS

Redirecting STDOUT and STDERR

Terminal — bash — 84×24

```
$ echo “Redirect stdout to a file” >file.text
$ cat file.text
Redirect stdout to a file
$ echo “Redirect and append stdout” >>file.text
$ cat file.txt
```

STREAMS

Redirecting STDOUT and STDERR

Terminal — bash — 84×24

```
$ echo “Redirect stdout to a file” >file.text
$ cat file.text
Redirect stdout to a file
$ echo “Redirect and append stdout” >>file.text
$ cat file.txt
Redirect stdout to a file
Redirect and append stdout
$
```

STREAMS

Redirecting STDOUT and STDERR

Terminal — bash — 84×24

```
$ echo “Redirect stdout to a file” >file.text
$ cat file.text
Redirect stdout to a file
$ echo “Redirect and append stdout” >>file.text
$ cat file.txt
Redirect stdout to a file
Redirect and append stdout
$
$ python “Redirect stderr to a file” 2>err.text
```

STREAMS

Redirecting STDOUT and STDERR

Terminal — bash — 84×24

```
$ echo “Redirect stdout to a file” >file.text
$ cat file.text
Redirect stdout to a file
$ echo “Redirect and append stdout” >>file.text
$ cat file.txt
Redirect stdout to a file
Redirect and append stdout
$
$ python “Redirect stderr to a file” 2>err.text
$
```

STREAMS

Redirecting STDOUT and STDERR

Terminal — bash — 84×24

```
$ echo “Redirect stdout to a file” >file.text
$ cat file.text
Redirect stdout to a file
$ echo “Redirect and append stdout” >>file.text
$ cat file.txt
Redirect stdout to a file
Redirect and append stdout
$
$ python “Redirect stderr to a file” 2>err.text
$ python “or append to a file” 2>>err.text
```

STREAMS

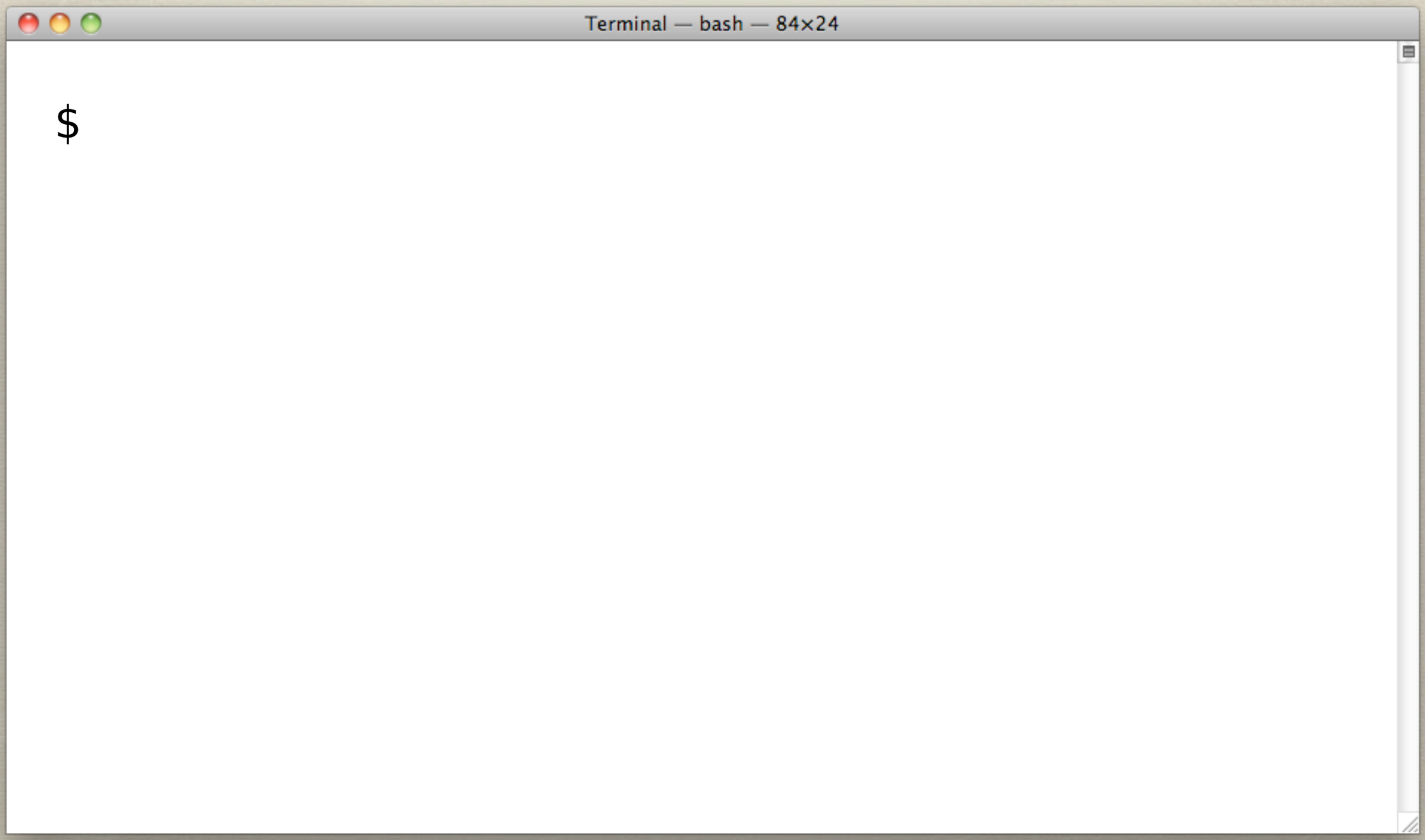
Redirecting STDOUT and STDERR

Terminal — bash — 84×24

```
$ echo “Redirect stdout to a file” >file.text
$ cat file.text
Redirect stdout to a file
$ echo “Redirect and append stdout” >>file.text
$ cat file.txt
Redirect stdout to a file
Redirect and append stdout
$
$ python “Redirect stderr to a file” 2>err.text
$ python “or append to a file” 2>>err.text
$
```

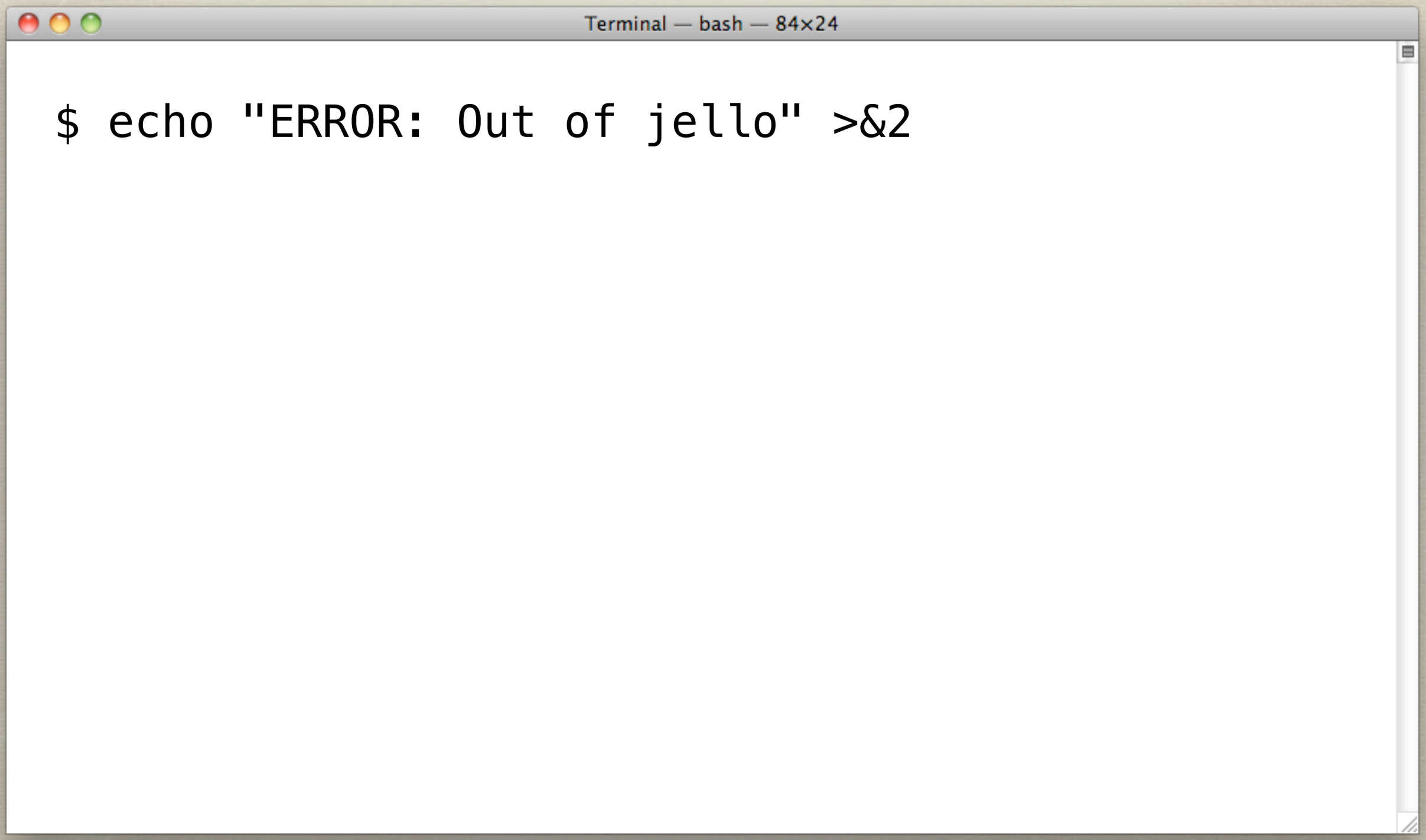
STREAMS

Redirecting to File Descriptors



STREAMS

Redirecting to File Descriptors



STREAMS

Redirecting to File Descriptors

Terminal — bash — 84×24

```
$ echo "ERROR: Out of jello" >&2
Error: Out of jello
$
```

STREAMS

Redirecting to File Descriptors

Terminal — bash — 84×24

```
$ echo "ERROR: Out of jello" >&2
Error: Out of jello
$
$
```

STREAMS

Redirecting to File Descriptors

Terminal — bash — 84×24

```
$ echo "ERROR: Out of jello" >&2
Error: Out of jello
$
$ exec 3> file.out   # like f=open() in python
$
```

STREAMS

Redirecting to File Descriptors

Terminal — bash — 84×24

```
$ echo "ERROR: Out of jello" >&2
Error: Out of jello
$
$ exec 3> file.out   # like f=open() in python
$ echo 'Armadillos!' >&3
$
```

STREAMS

Redirecting to File Descriptors

Terminal — bash — 84×24

```
$ echo "ERROR: Out of jello" >&2
Error: Out of jello
$
$ exec 3> file.out    # like f=open() in python
$ echo 'Armadillos!' >&3
$ 3>&-                # f.close in python
$
```

STREAMS

Redirecting to File Descriptors

Terminal — bash — 84×24

```
$ echo "ERROR: Out of jello" >&2
Error: Out of jello
$
$ exec 3> file.out    # like f=open() in python
$ echo 'Armadillos!' >&3
$ 3>&-                # f.close in python
$ cat file.out
```

STREAMS

Redirecting to File Descriptors

Terminal — bash — 84×24

```
$ echo "ERROR: Out of jello" >&2
Error: Out of jello
$
$ exec 3> file.out    # like f=open() in python
$ echo 'Armadillos!' >&3
$ 3>&-                # f.close in python
$ cat file.out
Armadillos!
$
```

56

PE UP
KS
s.
★

SS ★
g mat-
Wraps
olesale
)-[redacted]

50

1-operable, 2-fixed, low E
class, no grids, $750. pair, call
901-218-[redacted]

DRYER, KENMORE, $75

Heavy Duty, white, lg. capac-
ity. Runs great. (662) 449-[redacted]

FORK, mangled, $0.50.
Also selling garbage disposal,
used once, needs repair.
901-529-[redacted]

FREEZER (7.5 cubic feet) -
$60; Dinnerware - Harvest
Moon pattern, all white with
fruit, place setting for 12- $35;
Lamps - $5-$25; Side Tables-

WA[redacted]HER
c[redacted], su
more, wh
make offe

WASHER -
cost $1,
Call
★★★★★★

WICKER F
white, nig
bookcase,
ing, $325. E

WINE STO
Cedar in
pro[redacted]00 b
5[redacted]X38"W

PID = 1998

PID = 1998

PID = 2012

<Targets FD=34>

<Oracle FD=39>

<TeddyBear FD=41>

PID = 1998

PID = 2012

<Targets FD=34>

<Oracle FD=39>

<TeddyBear FD=41>

PID = 1998

PID = 2012

<Targets FD=34>

<Oracle FD=39>

<TeddyBear FD=41>

$ exec csi_duty.rb

PID = 1998

PID = 2012

<Targets FD=34>

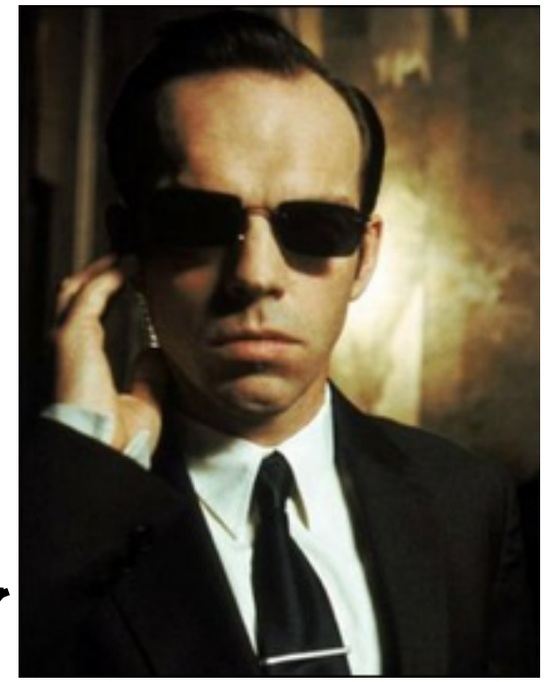
<Oracle FD=39>

<TeddyBear FD=41>

$ exec csi_duty.rb

PID = 1998

PID = 2012



PID = 1998

<Targets FD=34>

<Oracle FD=39>

<TeddyBear FD=41>

$ exec csi_duty.rb

PID = 2012

<InappropiateCommentsAboutTheBody FD=43>

PID = 1998

<Targets FD=34>

<Oracle FD=39>

<TeddyBear FD=41>

PID = 2012

PID = 1998

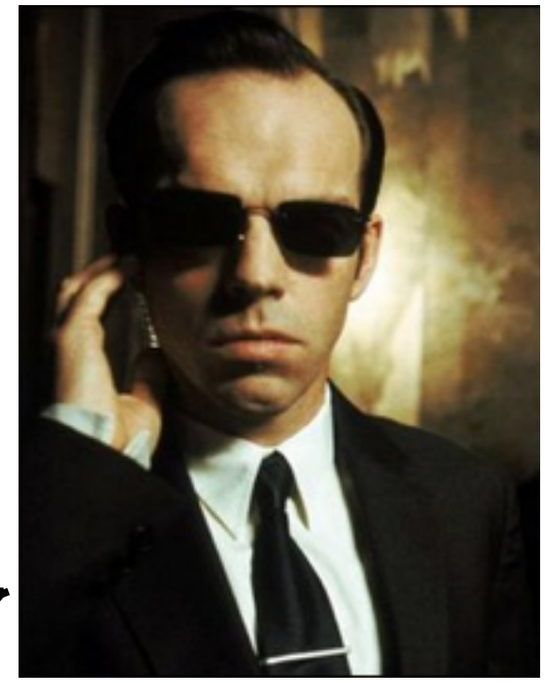
<Targets FD=34>

<Oracle FD=39>

<TeddyBear FD=41>

PID = 2012

$ wait 2012



PID = 1998

<Targets FD=34>

<Oracle FD=39>

<TeddyBear FD=41>

PID = 2012

$ wait 2012

PID = 1998

<Targets FD=34>

<Oracle FD=39>

<TeddyBear FD=41>

Parent PID 14724

Child PID 14725

```
$ ./fork_it.py
```

```
print "Parent pid is {0}".format(os.getpid())
```

Parent PID 14724

Child PID 14725

```
$ ./fork_it.py
```

```
    print "Parent pid is {0}".format(os.getpid())
▶   if (not os.fork()):
```

Parent PID 14724

Child PID 14725

```
$ ./fork_it.py
Parent pid is 14724
```

```
   print "Parent pid is {0}".format(os.getpid())
▶  if (not os.fork()):
```

Parent PID 14724

```
   print "Parent pid is {0}".format(os.getpid())
▶  if (not os.fork()):
```

Child PID 14725

```
$ ./fork_it.py
Parent pid is 14724
```

```
print "Parent pid is {0}".format(os.getpid())
if (not os.fork()):
    print "In child process. Pid is now {0}" \
        .format(os.getpid())
```

Parent PID 14724

```
print "Parent pid is {0}".format(os.getpid())
if (not os.fork()):
    print "In child process. Pid is now {0}" \
        .format(os.getpid())
```

Child PID 14725

```
$ ./fork_it.py
Parent pid is 14724
In child process. Pid is now 14725
```

```
print "Parent pid is {0}".format(os.getpid())
if (not os.fork()):
    print "In child process. Pid is now {0}" \
        .format(os.getpid())
    sys.exit(42)
```

Parent PID 14724

```
print "Parent pid is {0}".format(os.getpid())
if (not os.fork()):
    print "In child process. Pid is now {0}" \
        .format(os.getpid())
    sys.exit(42)
```

Child PID 14725

```
$ ./fork_it.py
Parent pid is 14724
In child process. Pid is now 14725
```

```
print "Parent pid is {0}".format(os.getpid())
if (not os.fork()):
    print "In child process. Pid is now {0}" \
        .format(os.getpid())
    sys.exit(42)
child_pid = Process.wait
```

Parent PID 14724

```
print "Parent pid is {0}".format(os.getpid())
if (not os.fork()):
    print "In child process. Pid is now {0}" \
        .format(os.getpid())
    sys.exit(42)
child_pid = Process.wait
```

Child PID 14725

```
$ ./fork_it.py
Parent pid is 14724
In child process. Pid is now 14725
```

```
print "Parent pid is {0}".format(os.getpid())
if (not os.fork()):
    print "In child process. Pid is now {0}" \
        .format(os.getpid())
    sys.exit(42)
child_pid = Process.wait
print "Child (pid {0}) terminated with status {1}" \
    .format(child_pid, exit_status >> 8)
```

Parent PID 14724

```
print "Parent pid is {0}".format(os.getpid())
if (not os.fork()):
    print "In child process. Pid is now {0}" \
        .format(os.getpid())
    sys.exit(42)
child_pid = Process.wait
print "Child (pid {0}) terminated with status {1}" \
    format(child_pid, exit_status >> 8)
```

Child PID 14725

```
$ ./fork_it.py
Parent pid is 14724
In child process. Pid is now 14725
Child (pid 14725) terminated with status 42
```

SOURCES

http://mij.oltrelinux.com/devel/unixprg/

http://www.faqs.org/docs/artu/

http://en.wikipedia.org/wiki/Pipeline_%28Unix%29

http://whynotwiki.com/Ruby_/_Process_management

http://www.unix.com/unix-dummies-questions-answers/100737-how-do-you-create-zombie-process.html

http://www.myelin.co.nz/post/2003/3/13/

http://cheezburger.com/

http://en.wikipedia.org/wiki/Unix_philosophy

http://vimeo.com/11202537

http://cheezburger.com/

https://github.com/gregmalcolm/unix_for_programmers_demo
https://gist.github.com/1241642
@gregmalcolm