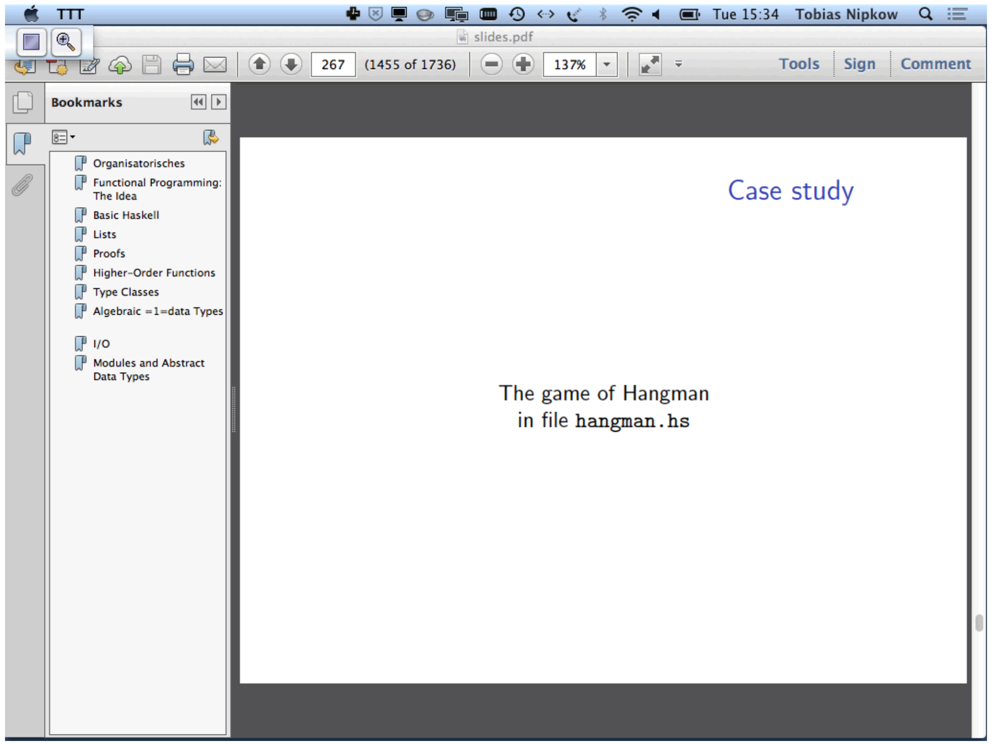
Script generated by TTT

Title: Nipkow: Info2 (17.12.2013)

Date: Tue Dec 17 15:34:31 CET 2013

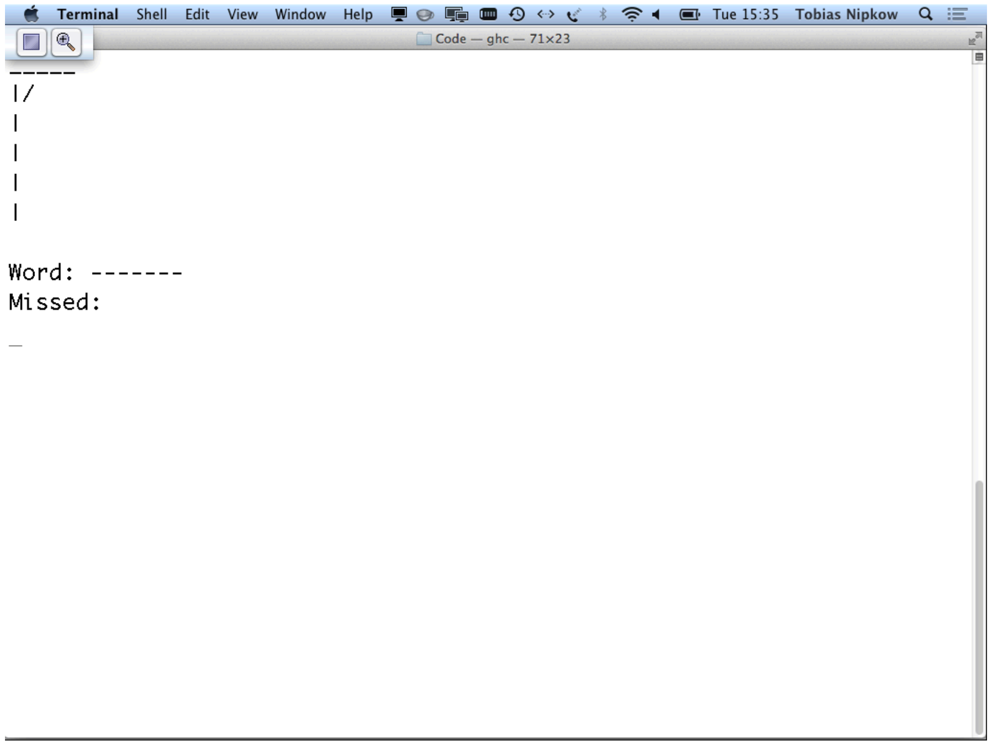
Duration: 67:28 min

Pages: 123

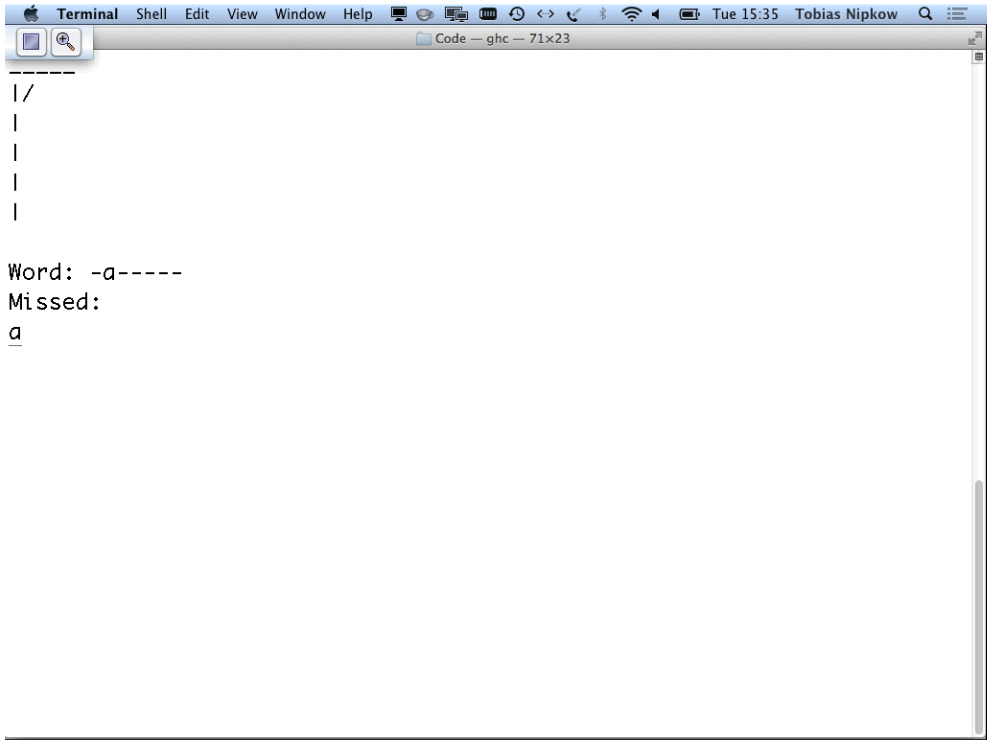


Case study

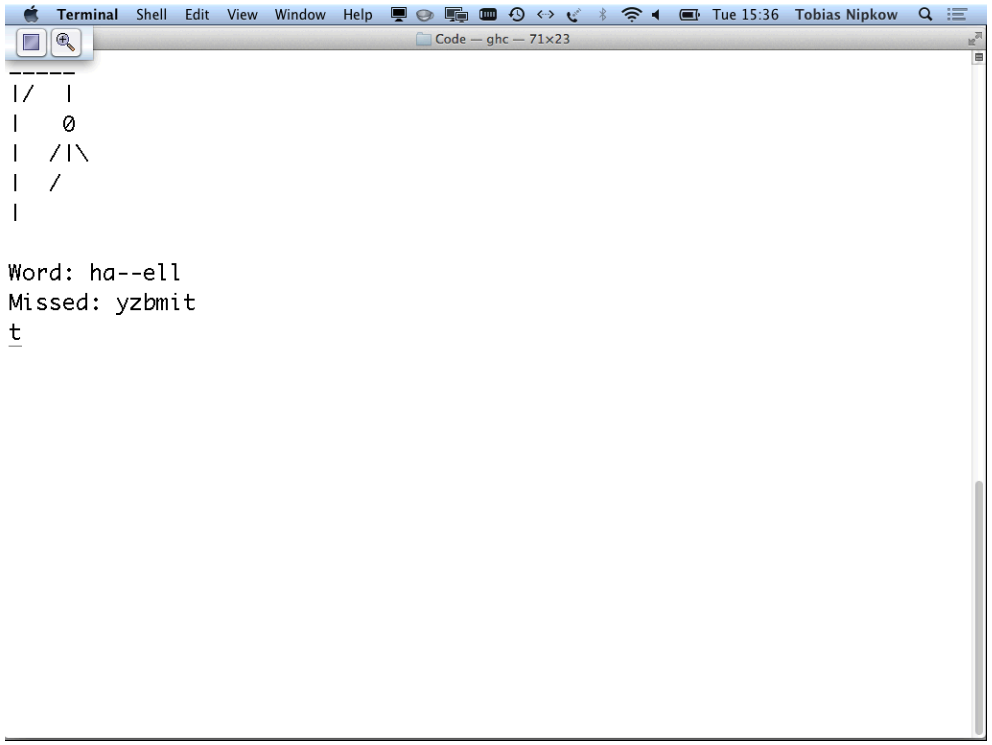
The game of Hangman
in file hangman.hs



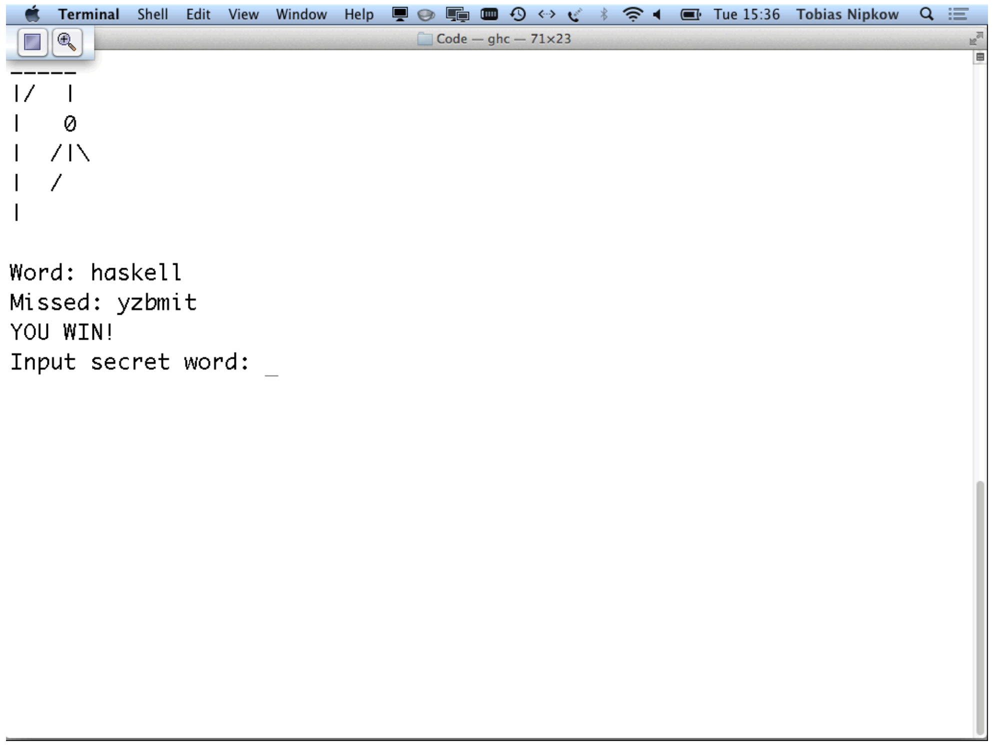
```
-----
|/
|
|
|
|

Word: -------
Missed:
```



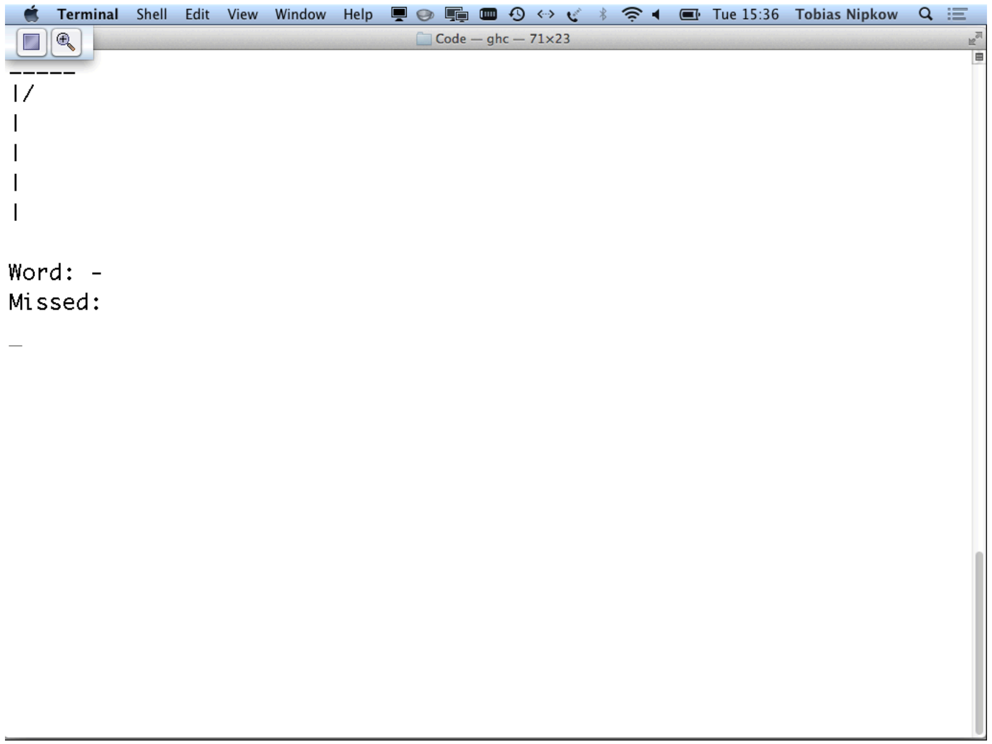
```
-----
|/
|
|
|
|

Word: -a-----
Missed:
a
```


Terminal Shell Edit View Window Help Tue 15:36 Tobias Nipkow
Code — ghc — 71×23

```
-----
|/  |
|   0
|  /|\
|  /
|

Word: ha--ell
Missed: yzbmit
t
```


Terminal Shell Edit View Window Help Tue 15:36 Tobias Nipkow
Code — ghc — 71×23

```
-----
|/  |
|   0
|  /|\
|  /
|

Word: haskell
Missed: yzbmit
YOU WIN!
Input secret word: _
```


Terminal Shell Edit View Window Help Tue 15:36 Tobias Nipkow
Code — ghc — 71×23

```
-----
|/
|
|
|
|

Word: -
Missed:

_
```

```
main :: IO ()
main = do putStr "Input secret word: "
```

```haskell
main :: IO ()
main = do putStr "Input secret word: "
          word <- getWord ""
```

```haskell
main :: IO ()
main = do putStr "Input secret word: "
          word <- getWord ""
          clear_screen
          guess word
          main
```

```haskell
guess :: String -> IO ()
```

```haskell
guess :: String -> IO ()
guess word = loop "" "" gallows
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word' =
             map (\x -> if x `elem` guessed
                        then x else '-')
               word
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word' =
             map (\x -> if x `elem` guessed
                        then x else '-')
               word
       writeAt (1,1)
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word' =
             map (\x -> if x `elem` guessed
                        then x else '-')
               word
       writeAt (1,1)
         (head gals ++ "\n"
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word' =
             map (\x -> if x `elem` guessed
                        then x else '-')
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word' ++
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word' =
             map (\x -> if x `elem` guessed
                        then x else '-')
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word' ++
          "\nMissed: " ++ missed ++ "\n")
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word' =
             map (\x -> if x `elem` guessed
                        then x else '-')
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word' ++
          "\nMissed: " ++ missed ++ "\n")
       if length gals == 1
```

```
uess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word’ =
             map (\x -> if x ‘elem‘ guessed
                        then x else ’-’)
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word’ ++
          "\nMissed: " ++ missed ++ "\n")
       if length gals == 1
       then putStrLn ("YOU ARE DEAD: " ++ word)
```

```
uess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word’ =
             map (\x -> if x ‘elem‘ guessed
                        then x else ’-’)
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word’ ++
          "\nMissed: " ++ missed ++ "\n")
       if length gals == 1
       then putStrLn ("YOU ARE DEAD: " ++ word)
       else if word’ == word then putStrLn "YOU WIN!"
```

```
uess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word’ =
             map (\x -> if x ‘elem‘ guessed
                        then x else ’-’)
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word’ ++
          "\nMissed: " ++ missed ++ "\n")
       if length gals == 1
       then putStrLn ("YOU ARE DEAD: " ++ word)
       else if word’ == word then putStrLn "YOU WIN!"
       else do c <- getChar
```

```
uess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word’ =
             map (\x -> if x ‘elem‘ guessed
                        then x else ’-’)
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word’ ++
          "\nMissed: " ++ missed ++ "\n")
       if length gals == 1
       then putStrLn ("YOU ARE DEAD: " ++ word)
       else if word’ == word then putStrLn "YOU WIN!"
       else do c <- getChar
               let ok = c ‘elem‘ word
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word' =
             map (\x -> if x `elem` guessed
                        then x else '-')
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word' ++
          "\nMissed: " ++ missed ++ "\n")
       if length gals == 1
       then putStrLn ("YOU ARE DEAD: " ++ word)
       else if word' == word then putStrLn "YOU WIN!"
       else do c <- getChar
               let ok = c `elem` word
               loop (if ok then c:guessed else guessed)
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word' =
             map (\x -> if x `elem` guessed
                        then x else '-')
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word' ++
          "\nMissed: " ++ missed ++ "\n")
       if length gals == 1
       then putStrLn ("YOU ARE DEAD: " ++ word)
       else if word' == word then putStrLn "YOU WIN!"
       else do c <- getChar
               let ok = c `elem` word
               loop (if ok then c:guessed else guessed)
                    (if ok then missed else missed++[c])
```

```
guess :: String -> IO ()
guess word = loop "" "" gallows  where
  loop :: String -> String -> [String] -> IO()
  loop guessed missed gals =
    do let word' =
             map (\x -> if x `elem` guessed
                        then x else '-')
               word
       writeAt (1,1)
         (head gals ++ "\n" ++ "Word: " ++ word' ++
          "\nMissed: " ++ missed ++ "\n")
       if length gals == 1
       then putStrLn ("YOU ARE DEAD: " ++ word)
       else if word' == word then putStrLn "YOU WIN!"
       else do c <- getChar
               let ok = c `elem` word
               loop (if ok then c:guessed else guessed)
                    (if ok then missed else missed++[c])
                    (if ok then gals else tail gals)
```

Once IO, always IO

Once IO, always IO

You cannot add I/O to a function without giving it an IO type

Once IO, always IO

You cannot add I/O to a function without giving it an IO type

For example

```
sq :: Int -> Int        cube :: Int -> Int
sq x = x*x              cube x = x * sq x
```

Once IO, always IO

You cannot add I/O to a function without giving it an IO type

For example

```
sq :: Int -> Int        cube :: Int -> Int
sq x = x*x              cube x = x * sq x
```

Let us try to make sq print out some message:

```
sq x = do putStr("I am in sq!")
          return(x*x)
```

Once IO, always IO

You cannot add I/O to a function without giving it an IO type

For example

```
sq :: Int -> Int        cube :: Int -> Int
sq x = x*x              cube x = x * sq x
```

Let us try to make sq print out some message:

```
sq x = do putStr("I am in sq!")
          return(x*x)
```

What is the type of sq now? `Int -> IO Int`

Once IO, always IO

You cannot add I/O to a function without giving it an IO type

For example

```
sq :: Int -> Int        cube :: Int -> Int
sq x = x*x              cube x = x * sq x
```

Let us try to make sq print out some message:

```
sq x = do putStr("I am in sq!")
          return(x*x)
```

What is the type of sq now? `Int -> IO Int`
And this is what happens to cube:

```
cube x = do x2 <- sq x
            return(x * x2)
```

Haskell is a pure functional language

Separate I/O from processing to reduce IO creep:

Separate I/O from processing to reduce IO creep:

```
main :: IO ()
main = do s <- getLine
          let r = process s
          putStrLn r
          main
```

Separate I/O from processing to reduce IO creep:

```
main :: IO ()
main = do s <- getLine
          let r = process s
          putStrLn r
          main

process :: String -> String
process s = ...
```

Separate I/O from processing to reduce IO creep:

```
main :: IO ()
main = do s <- getLine
          let r = process s
          putStrLn r
          main

process :: String -> String
process s = ...
```

9.1 File I/O

The simple way

- `type FilePath = String`
- `readFile :: FilePath -> IO String`

The simple way

- `type FilePath = String`
- `readFile :: FilePath -> IO String`

  Reads file contents *lazily*,

The simple way

- `type FilePath = String`
- `readFile :: FilePath -> IO String`

  Reads file contents *lazily*,
  only as much as is needed
- `writeFile :: FilePath -> String -> IO ()`

The simple way

- `type FilePath = String`
- `readFile :: FilePath -> IO String`

  Reads file contents *lazily*,
  only as much as is needed
- `writeFile :: FilePath -> String -> IO ()`

  Writes whole file
- `appendFile :: FilePath -> String -> IO ()`

Handles

```
data Handle
```

Handles

`data Handle`

Opaque type, implementation dependent

> *Haskell defines operations to read and write characters from and to files, represented by values of type* `Handle`.

```
import System.IO
```

Files and handles

-
  ```
  data IOMode = ReadMode | WriteMode
              | AppendMode | ReadWriteMode
  ```

Handles

`data Handle`

Opaque type, implementation dependent

Files and handles

- `data IOMode = ReadMode | WriteMode`
  `             | AppendMode | ReadWriteMode`
- `openFile :: FilePath -> IOMode -> IO Handle`

  Creates handle to file and opens file
- `hClose :: Handle -> IO ()`

By convention
all IO actions that take a handle argument begin with `h`

In ReadMode

- `hGetChar :: Handle -> IO Char`

In ReadMode

- `hGetChar :: Handle -> IO Char`
- `hGetLine :: Handle -> IO String`
- `hGetContents :: Handle -> IO String`

  Reads the whole file *lazily*

In WriteMode

- `hPutChar :: Handle -> Char -> IO ()`

In WriteMode

- `hPutChar :: Handle -> Char -> IO ()`
- `hPutStr :: Handle -> String -> IO ()`
- `hPutStrLn :: Handle -> String -> IO ()`
- `hPrint :: Show a => Handle -> a -> IO ()`

stdin and stdout

- `stdin :: Handle`
  `stdout :: Handle`

stdin and stdout

- `stdin :: Handle`
  `stdout :: Handle`
- `getChar = hGetChar stdin`
  `putChar = hPutChar stdout`

There is much more in the Standard IO Library

There is much more in the Standard IO Library
(including exception handling for IO actions)

Example (interactive cp: icp.hs)

```
main :: IO()
```

Example (interactive cp: icp.hs)

```
main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
```

Example (interactive cp: icp.hs)

```
main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
     toH <- readOpenFile "Copy to: " WriteMode
```

Example (interactive cp: icp.hs)

```
main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
     toH <- readOpenFile "Copy to: " WriteMode
     contents <- hGetContents fromH
     hPutStr toH contents
```

Example (interactive cp: icp.hs)

```
main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
     toH <- readOpenFile "Copy to: " WriteMode
     contents <- hGetContents fromH
     hPutStr toH contents
     hClose fromH
     hClose toH

readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
```

Example (interactive cp: icp.hs)

```
main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
     toH <- readOpenFile "Copy to: " WriteMode
     contents <- hGetContents fromH
     hPutStr toH contents
     hClose fromH
     hClose toH

readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
  do putStrLn prompt
```

Example (interactive cp: icp.hs)

```
main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
     toH <- readOpenFile "Copy to: " WriteMode
     contents <- hGetContents fromH
     hPutStr toH contents
     hClose fromH
     hClose toH

readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
  do putStrLn prompt
     name <- getLine
     handle <- openFile name mode
```

Example (interactive cp: icp.hs)

```
main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
     toH <- readOpenFile "Copy to: " WriteMode
     contents <- hGetContents fromH
     hPutStr toH contents
     hClose fromH
     hClose toH

readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
  do putStrLn prompt
     name <- getLine
     handle <- openFile name mode
     return handle
```

Executing `xyz.hs`

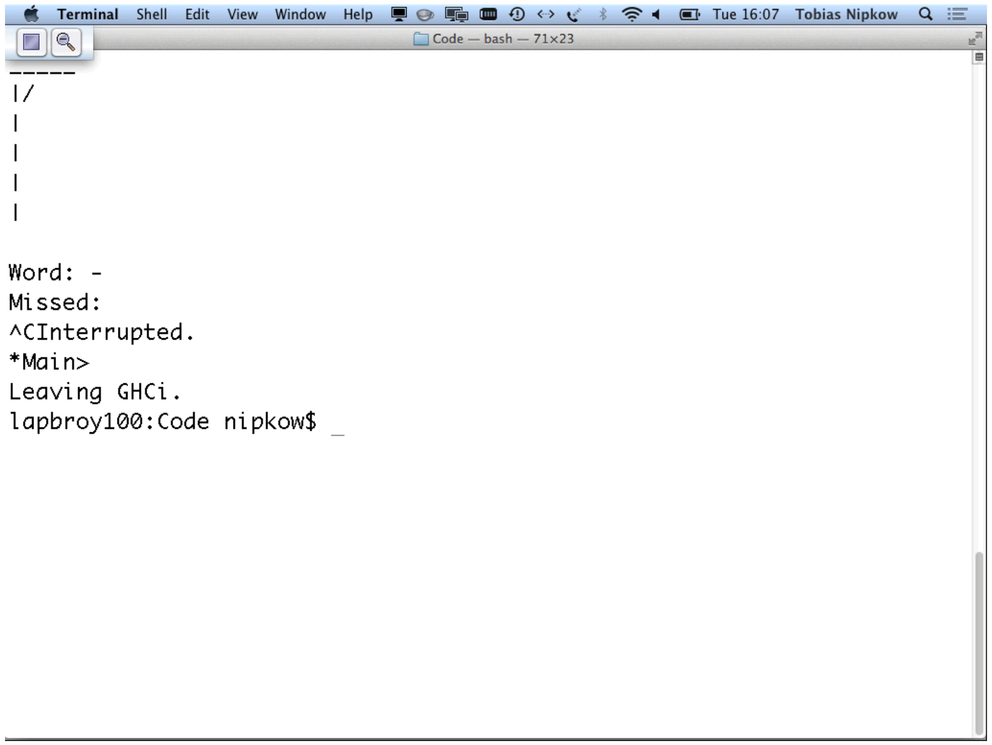
If `xyz.hs` contains a definition of `main`:

- `runhaskell xyz`

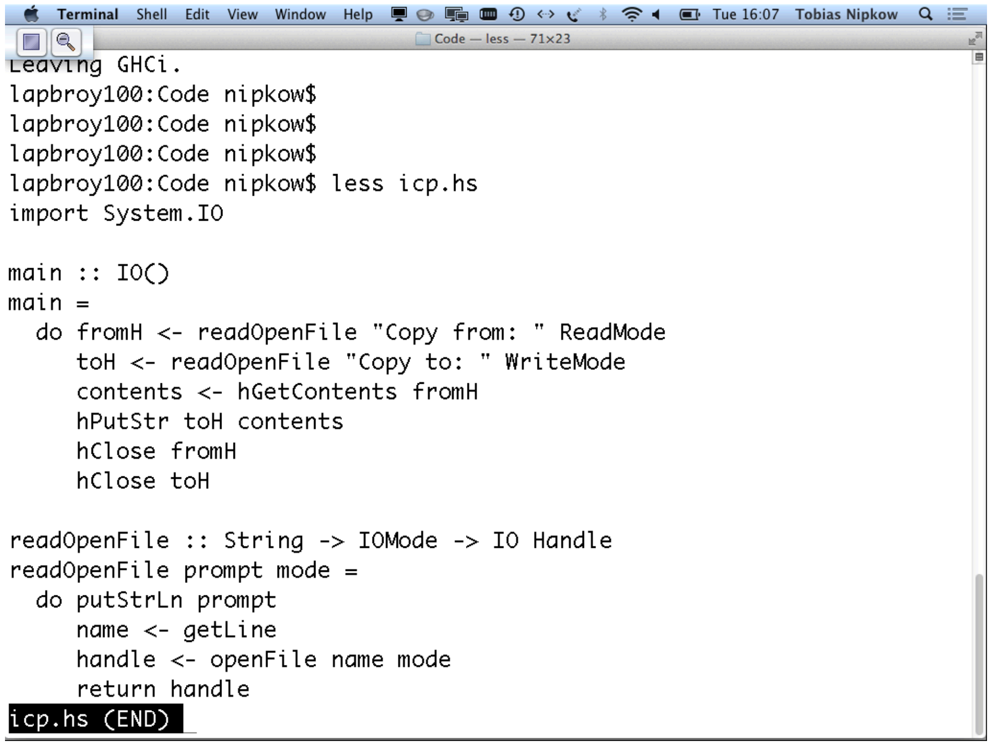
Executing `xyz.hs`

If `xyz.hs` contains a definition of `main`:

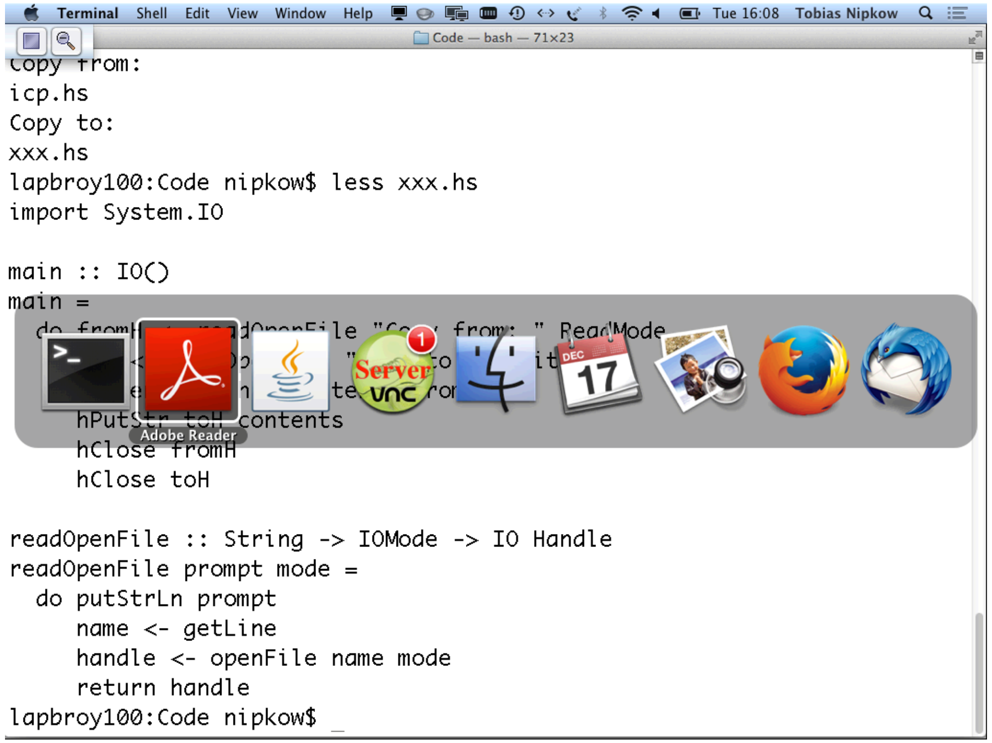
- `runhaskell xyz`
  or
- `ghc xyz` $\rightsquigarrow$ executable file `xyz`


Terminal Shell Edit View Window Help Tue 16:07 Tobias Nipkow
Code — bash — 71×23

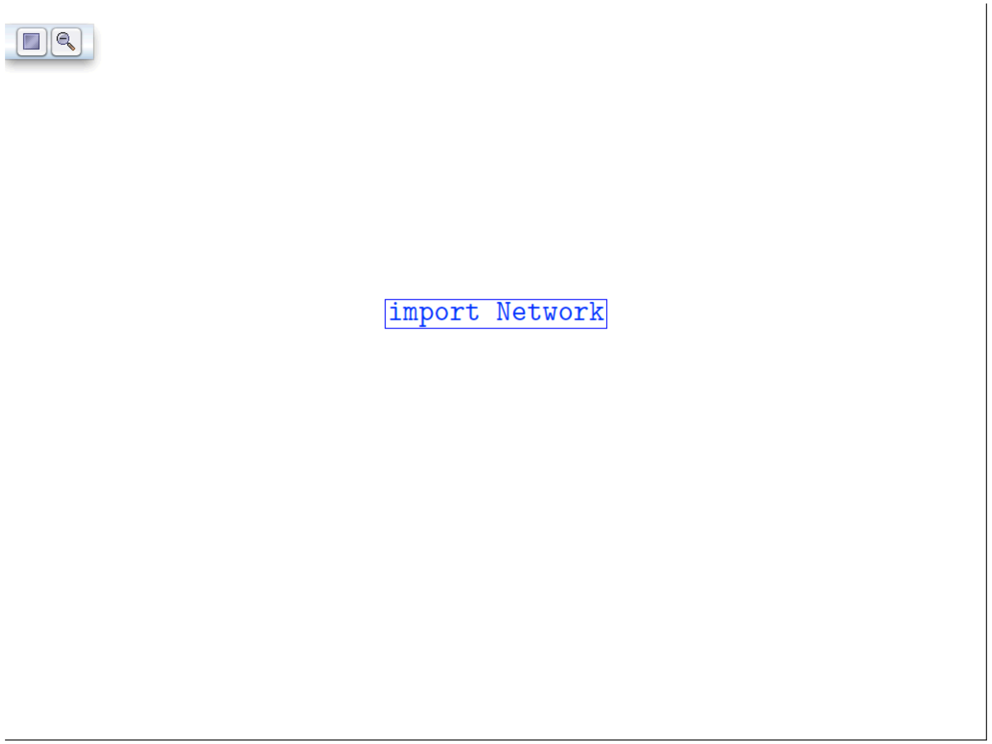
```
-----
|/
|
|
|
|

Word: -
Missed:
^CInterrupted.
*Main>
Leaving GHCi.
lapbroy100:Code nipkow$ _
```


Terminal Shell Edit View Window Help Tue 16:07 Tobias Nipkow
Code — less — 71×23

```
Leaving GHCi.
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ less icp.hs
import System.IO

main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
     toH <- readOpenFile "Copy to: " WriteMode
     contents <- hGetContents fromH
     hPutStr toH contents
     hClose fromH
     hClose toH

readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
  do putStrLn prompt
     name <- getLine
     handle <- openFile name mode
     return handle
icp.hs (END)
```


Terminal Shell Edit View Window Help Tue 16:08 Tobias Nipkow
Code — bash — 71×23
Adobe Reader

```
Copy from:
icp.hs
Copy to:
xxx.hs
lapbroy100:Code nipkow$ less xxx.hs
import System.IO

main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
     hPutStr toH contents
     hClose fromH
     hClose toH

readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
  do putStrLn prompt
     name <- getLine
     handle <- openFile name mode
     return handle
lapbroy100:Code nipkow$ _
```



```
import Network
```

Types

- `data Socket`
- `data PortId = PortNumber PortNumber | ...`

Types

- `data Socket`
- `data PortId = PortNumber PortNumber | ...`
- `data PortNumber`
  `instance Num PortNumber`

Types

- `data Socket`
- `data PortId = PortNumber PortNumber | ...`
- `data PortNumber`
  `instance Num PortNumber`
  $\Longrightarrow$ `PortNumber 9000 :: PortId`

Server functions

- `listenOn :: PortId -> IO Socket`
  Create server side socket for specific port

Server functions

- `listenOn :: PortId -> IO Socket`
  Create server side socket for specific port
- `accept :: Socket -> IO (Handle, ..., ...)`
  $\Longrightarrow$ can read/write from/to socket via handle

Server functions

- `listenOn :: PortId -> IO Socket`
  Create server side socket for specific port
- `accept :: Socket -> IO (Handle, ..., ...)`
  $\Longrightarrow$ can read/write from/to socket via handle
- `sClose :: Socket -> IO ()`
  Close socket

Initialization for Windows

```
withSocketsDo :: IO a -> IO a
```

Initialization for Windows

```
withSocketsDo :: IO a -> IO a
```

Standard use pattern:

```
main  =  withSocketsDo $ do ...
```

Initialization for Windows

```
withSocketsDo :: IO a -> IO a
```

Standard use pattern:

```
main  =  withSocketsDo $ do ...
```

Does nothing under Unix

Example (pingPong.hs)

Example (pingPong.hs)

```
main :: IO ()
main  =  withSocketsDo $ do
  sock <- listenOn $ PortNumber 9000
```

Initialization for Windows

```
withSocketsDo :: IO a -> IO a
```

Standard use pattern:

```
main  =  withSocketsDo $ do ...
```

Does nothing under Unix

Example (pingPong.hs)

```
main :: IO ()
main  =  withSocketsDo $ do
  sock <- listenOn $ PortNumber 9000
  (h, _, _) <- accept sock
  hSetBuffering h LineBuffering
```

Example (pingPong.hs)

```
main :: IO ()
main  =  withSocketsDo $ do
  sock <- listenOn $ PortNumber 9000
  (h, _, _) <- accept sock
  hSetBuffering h LineBuffering
  loop h
```

Example (pingPong.hs)

```
main :: IO ()
main  =  withSocketsDo $ do
  sock <- listenOn $ PortNumber 9000
  (h, _, _) <- accept sock
  hSetBuffering h LineBuffering
  loop h
  sClose sock
```

Example (pingPong.hs)

```
main :: IO ()
main  =  withSocketsDo $ do
  sock <- listenOn $ PortNumber 9000
  (h, _, _) <- accept sock
  hSetBuffering h LineBuffering
  loop h
  sClose sock

loop :: Handle -> IO ()
loop h  =  do
  input <- hGetLine h
  if take 4 input == "quit"
```

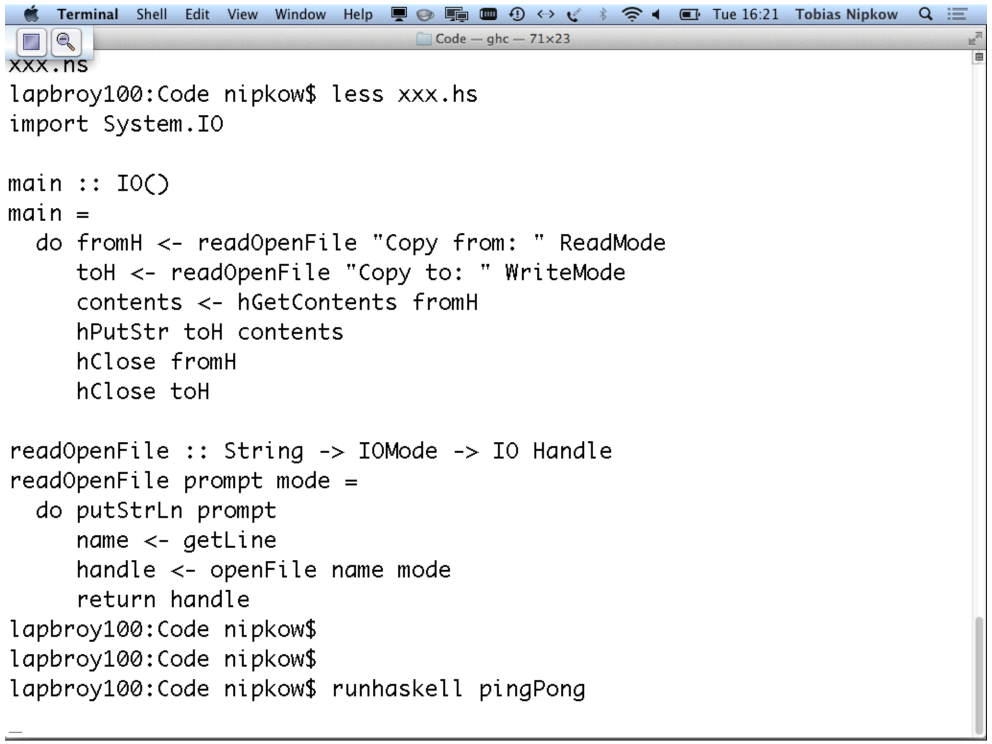
Example (pingPong.hs)

```haskell
main :: IO ()
main  =  withSocketsDo $ do
  sock <- listenOn $ PortNumber 9000
  (h, _, _) <- accept sock
  hSetBuffering h LineBuffering
  loop h
  sClose sock

loop :: Handle -> IO ()
loop h  =  do
  input <- hGetLine h
  if take 4 input == "quit"
  then do hPutStrLn h "goodbye!"
          hClose h
```

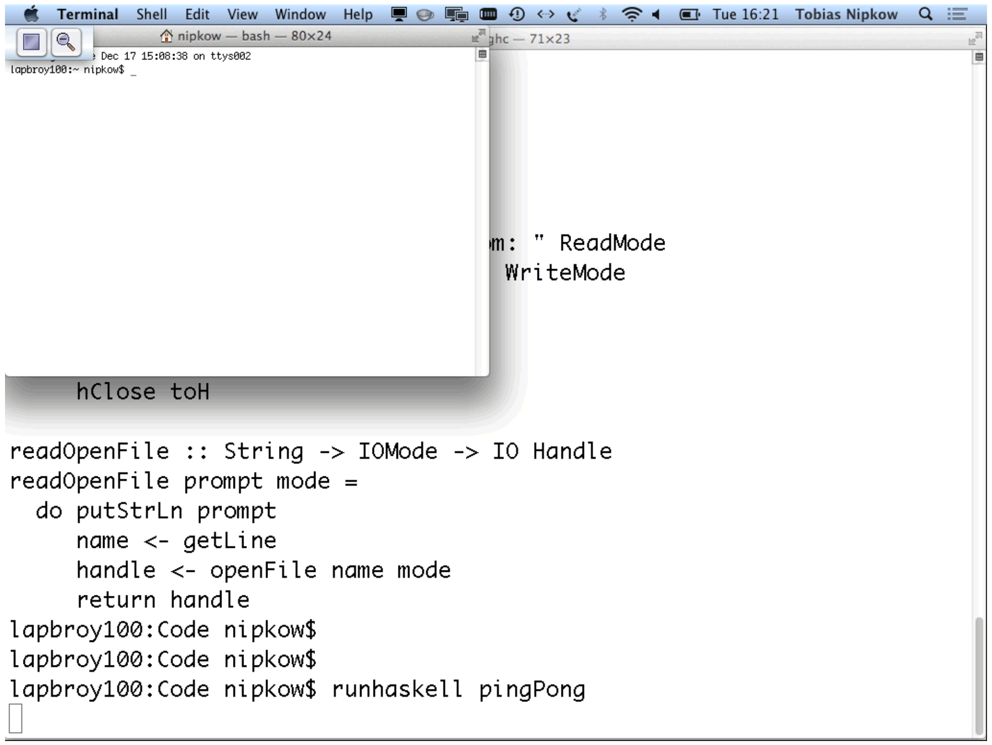
Example (pingPong.hs)

```haskell
main :: IO ()
main  =  withSocketsDo $ do
  sock <- listenOn $ PortNumber 9000
  (h, _, _) <- accept sock
  hSetBuffering h LineBuffering
  loop h
  sClose sock

loop :: Handle -> IO ()
loop h  =  do
  input <- hGetLine h
  if take 4 input == "quit"
  then do hPutStrLn h "goodbye!"
          hClose h
  else do hPutStrLn h ("got " ++ input)
          loop h
```

Client functions

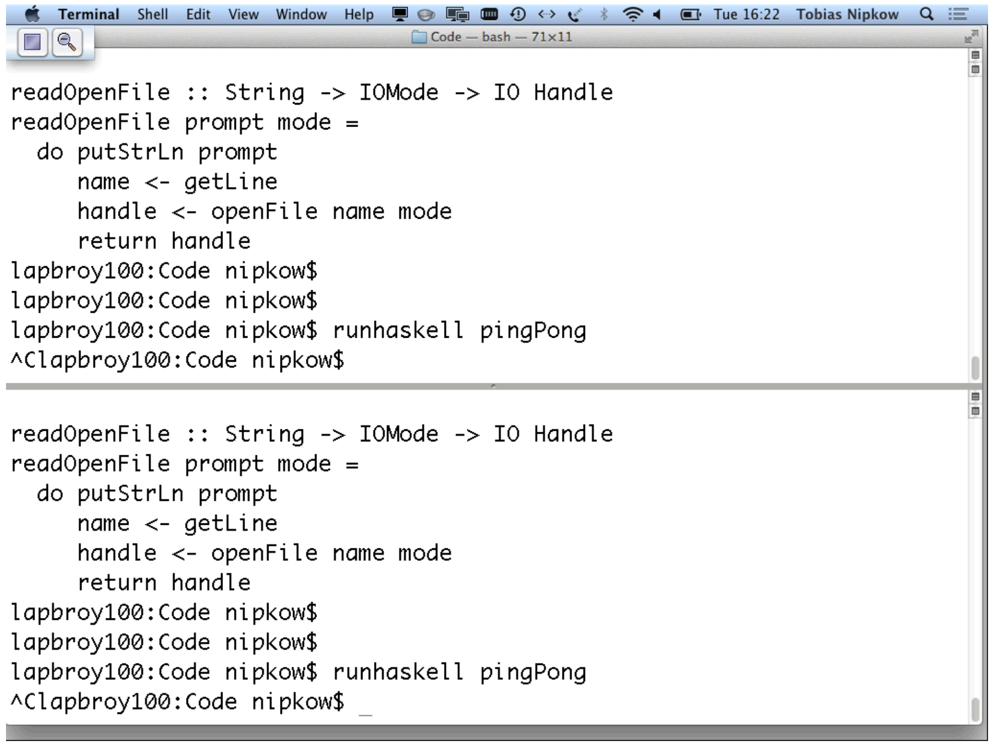


```
xxx.hs
lapbroy100:Code nipkow$ less xxx.hs
import System.IO

main :: IO()
main =
  do fromH <- readOpenFile "Copy from: " ReadMode
     toH <- readOpenFile "Copy to: " WriteMode
     contents <- hGetContents fromH
     hPutStr toH contents
     hClose fromH
     hClose toH

readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
  do putStrLn prompt
     name <- getLine
     handle <- openFile name mode
     return handle
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
```


Terminal Shell Edit View Window Help Tue 16:21 Tobias Nipkow
nipkow — bash — 80×24
ghc — 71×23

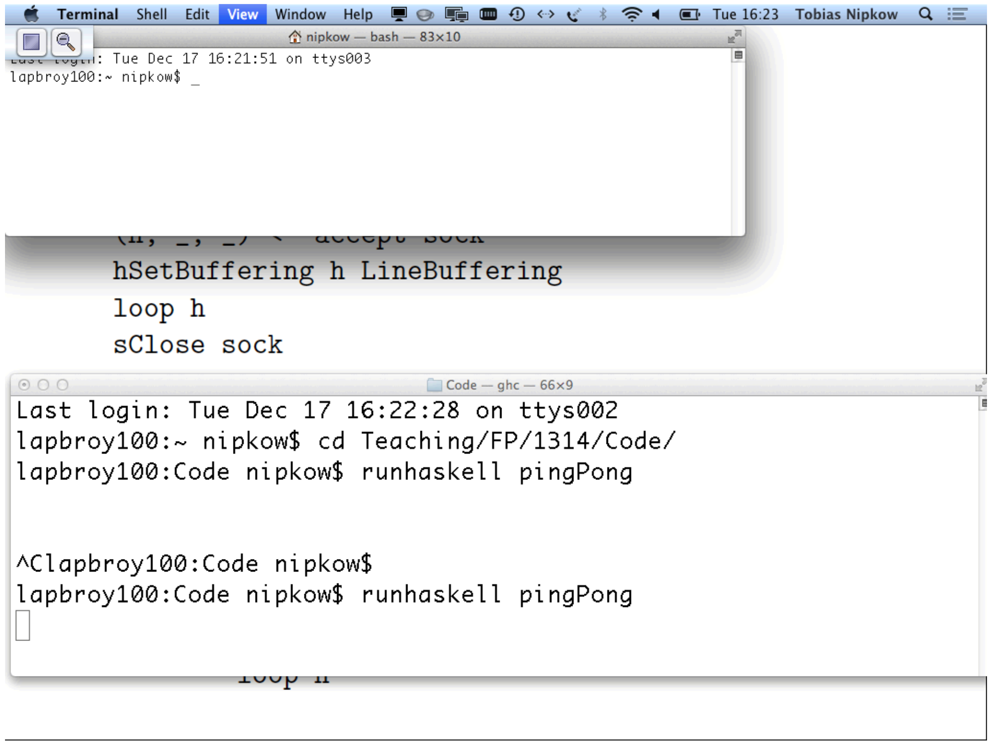
```
Dec 17 15:08:38 on ttys002
lapbroy100:~ nipkow$ _
```

```
m: " ReadMode
 WriteMode

    hClose toH

readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
  do putStrLn prompt
     name <- getLine
     handle <- openFile name mode
     return handle
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
```


Terminal Shell Edit View Window Help Tue 16:22 Tobias Nipkow
Code — bash — 71×11

```
readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
  do putStrLn prompt
     name <- getLine
     handle <- openFile name mode
     return handle
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
^Clapbroy100:Code nipkow$
```

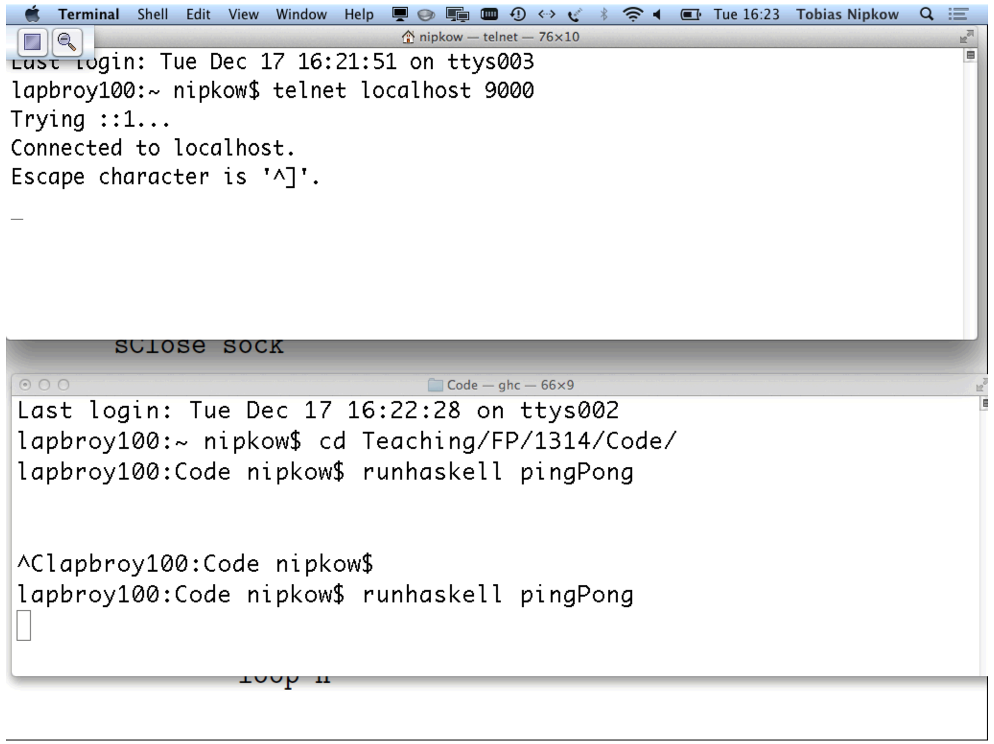
```
readOpenFile :: String -> IOMode -> IO Handle
readOpenFile prompt mode =
  do putStrLn prompt
     name <- getLine
     handle <- openFile name mode
     return handle
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
^Clapbroy100:Code nipkow$ _
```


Terminal Shell Edit View Window Help Tue 16:23 Tobias Nipkow
nipkow — bash — 83×10
Code — ghc — 66×9

```
Last login: Tue Dec 17 16:21:51 on ttys003
lapbroy100:~ nipkow$ _
```

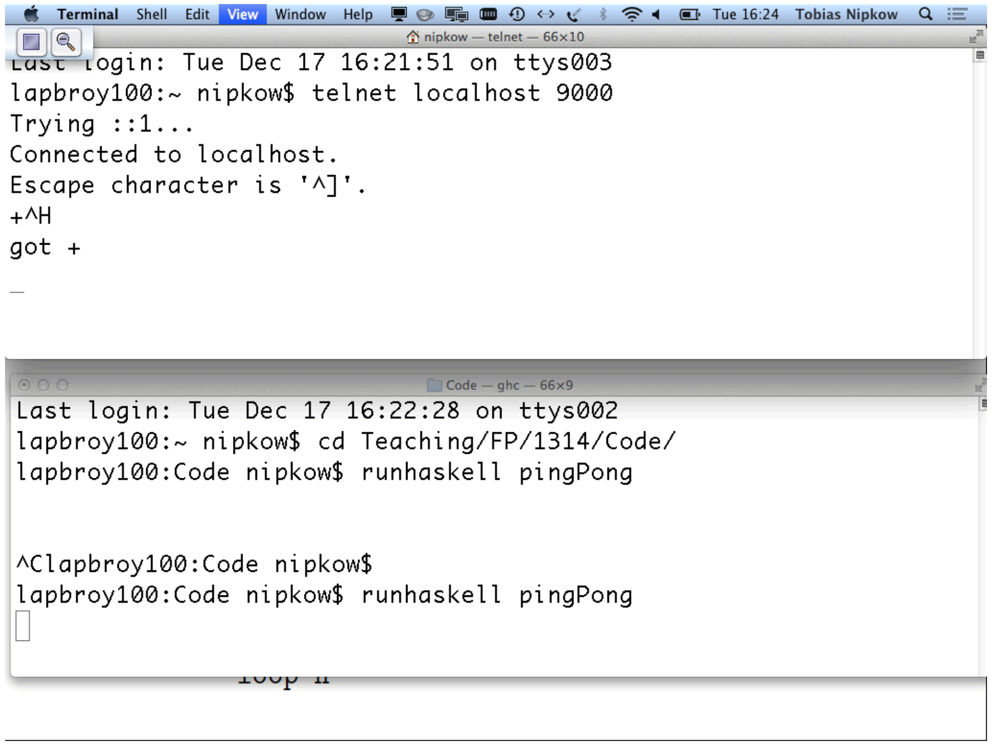
```
    (h, _, _) <- accept sock
    hSetBuffering h LineBuffering
    loop h
    sClose sock
```

```
Last login: Tue Dec 17 16:22:28 on ttys002
lapbroy100:~ nipkow$ cd Teaching/FP/1314/Code/
lapbroy100:Code nipkow$ runhaskell pingPong


^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
```


Terminal Shell Edit View Window Help Tue 16:23 Tobias Nipkow
nipkow — telnet — 76×10
Code — ghc — 66×9

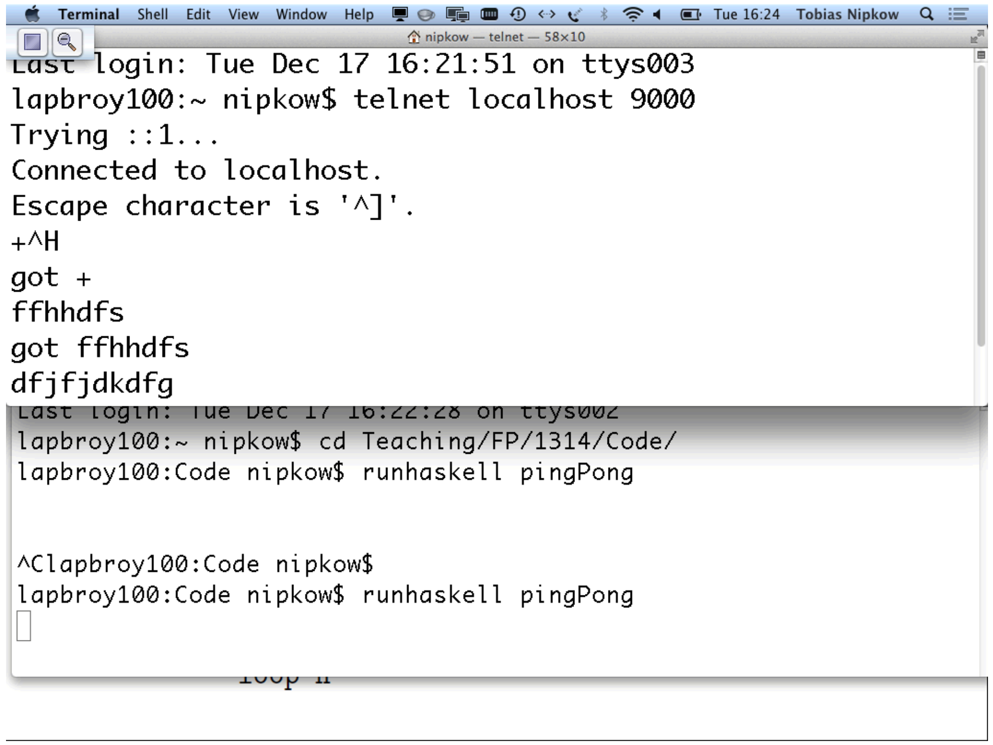
```
Last login: Tue Dec 17 16:21:51 on ttys003
lapbroy100:~ nipkow$ telnet localhost 9000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
_
```

```
Last login: Tue Dec 17 16:22:28 on ttys002
lapbroy100:~ nipkow$ cd Teaching/FP/1314/Code/
lapbroy100:Code nipkow$ runhaskell pingPong


^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
```



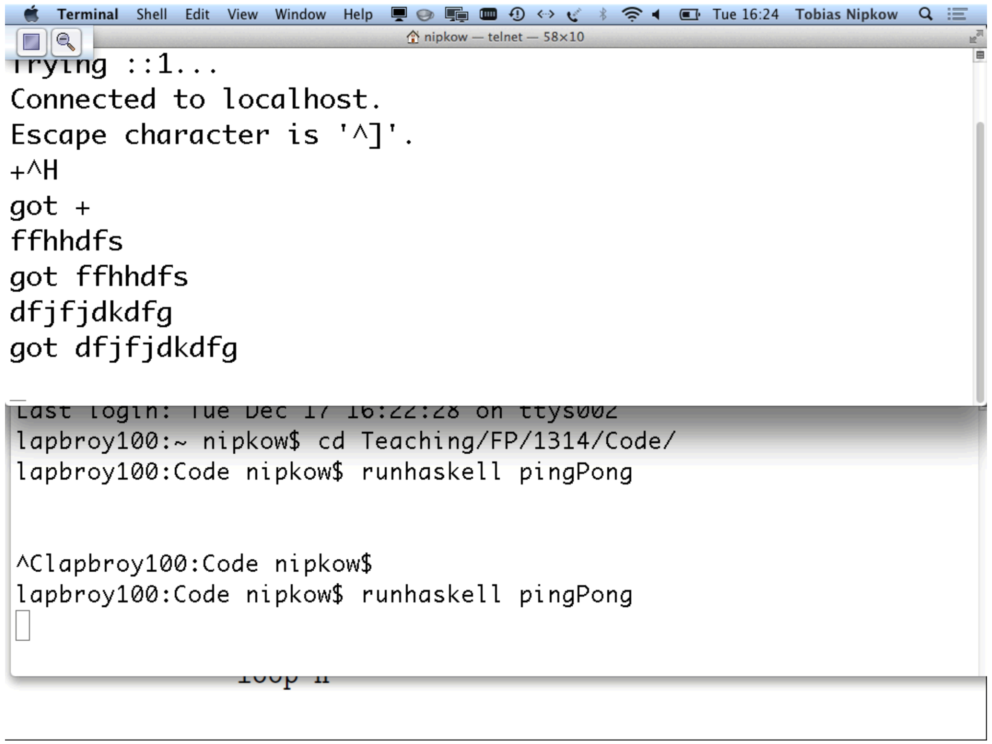
```
Last login: Tue Dec 17 16:21:51 on ttys003
lapbroy100:~ nipkow$ telnet localhost 9000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
+^H
got +
```

```
Last login: Tue Dec 17 16:22:28 on ttys002
lapbroy100:~ nipkow$ cd Teaching/FP/1314/Code/
lapbroy100:Code nipkow$ runhaskell pingPong


^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
```



```
Last login: Tue Dec 17 16:21:51 on ttys003
lapbroy100:~ nipkow$ telnet localhost 9000
Trying ::1...
Connected to localhost.
Escape character is '^]'.
+^H
got +
ffhhdfs
got ffhhdfs
dfjfjdkdfg
```

```
Last login: Tue Dec 17 16:22:28 on ttys002
lapbroy100:~ nipkow$ cd Teaching/FP/1314/Code/
lapbroy100:Code nipkow$ runhaskell pingPong


^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
```



```
Trying ::1...
Connected to localhost.
Escape character is '^]'.
+^H
got +
ffhhdfs
got ffhhdfs
dfjfjdkdfg
got dfjfjdkdfg
```

```
Last login: Tue Dec 17 16:22:28 on ttys002
lapbroy100:~ nipkow$ cd Teaching/FP/1314/Code/
lapbroy100:Code nipkow$ runhaskell pingPong


^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
```

Client functions

Client functions

- `type HostName = String`
  For example `"haskell.org"` or `"192.168.0.1"`

Client functions

- `type HostName = String`
  For example `"haskell.org"` or `"192.168.0.1"`
- `connectTo :: HostName -> PortId -> IO Handle`
  Connect to specific port of specific host

Example (wGet.hs)

Example (wGet.hs)

```
main :: IO()
main = withSocketsDo $ do
```

Example (wGet.hs)

```
main :: IO()
main = withSocketsDo $ do
  putStrLn "Host?"
  host <- getLine
```

Example (wGet.hs)

```
main :: IO()
main = withSocketsDo $ do
  putStrLn "Host?"
  host <- getLine
  h <- connectTo host (PortNumber 80)
```

Example (wGet.hs)

```
main :: IO()
main = withSocketsDo $ do
  putStrLn "Host?"
  host <- getLine
  h <- connectTo host (PortNumber 80)
  hSetBuffering h LineBuffering
  putStrLn "Resource?"
  res <- getLine
```

Example (wGet.hs)

```
main :: IO()
main = withSocketsDo $ do
  putStrLn "Host?"
  host <- getLine
  h <- connectTo host (PortNumber 80)
  hSetBuffering h LineBuffering
  putStrLn "Resource?"
  res <- getLine
  hPutStrLn h ("GET " ++ res ++ " HTTP/1.0\n")
```

Example (wGet.hs)

```
main :: IO()
main = withSocketsDo $ do
  putStrLn "Host?"
  host <- getLine
  h <- connectTo host (PortNumber 80)
  hSetBuffering h LineBuffering
  putStrLn "Resource?"
  res <- getLine
  hPutStrLn h ("GET " ++ res ++ " HTTP/1.0\n")
  s <- hGetContents h
```

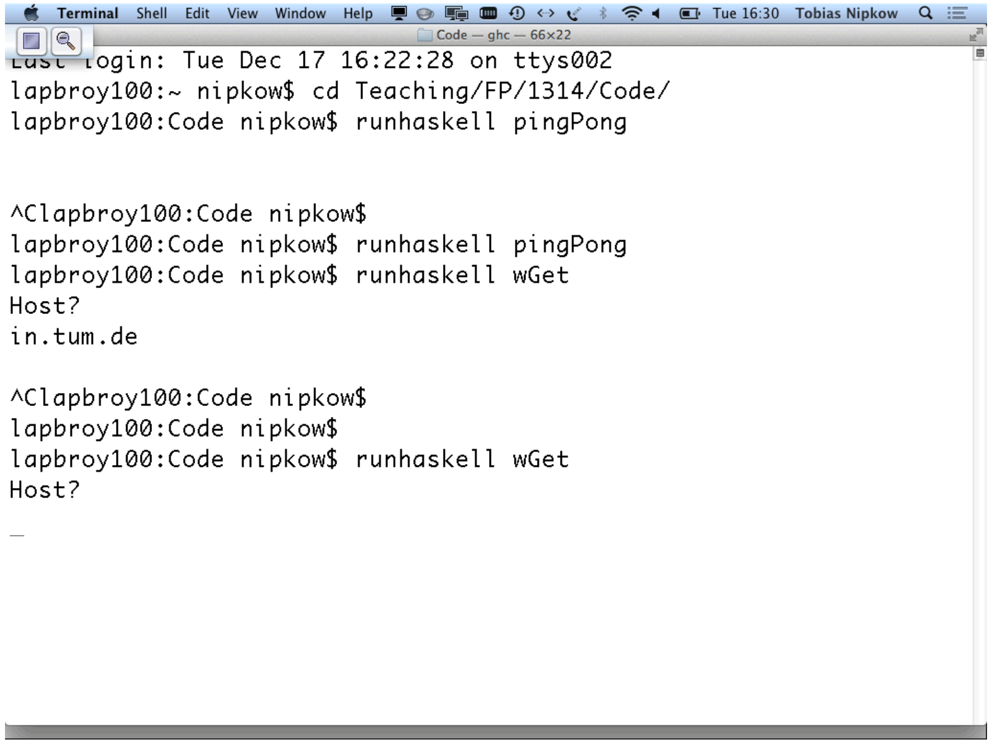
Example (wGet.hs)

```
main :: IO()
main = withSocketsDo $ do
  putStrLn "Host?"
  host <- getLine
  h <- connectTo host (PortNumber 80)
  hSetBuffering h LineBuffering
  putStrLn "Resource?"
  res <- getLine
  hPutStrLn h ("GET " ++ res ++ " HTTP/1.0\n")
  s <- hGetContents h
  putStrLn s
```

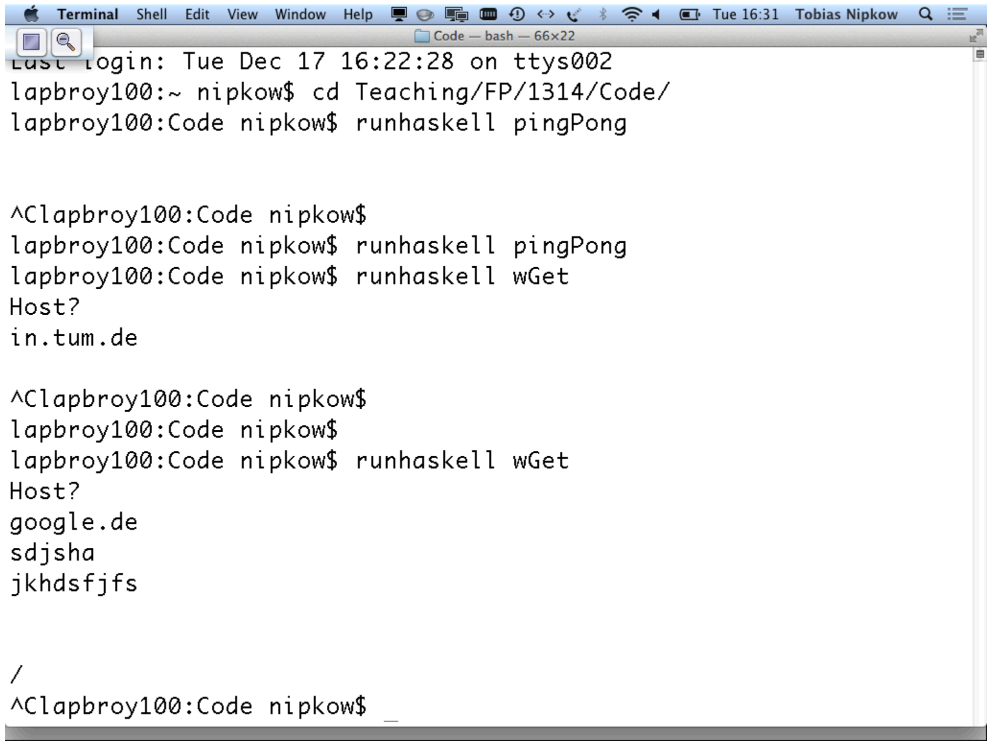
For more detail see

http://hackage.haskell.org/package/network/docs/Network.html

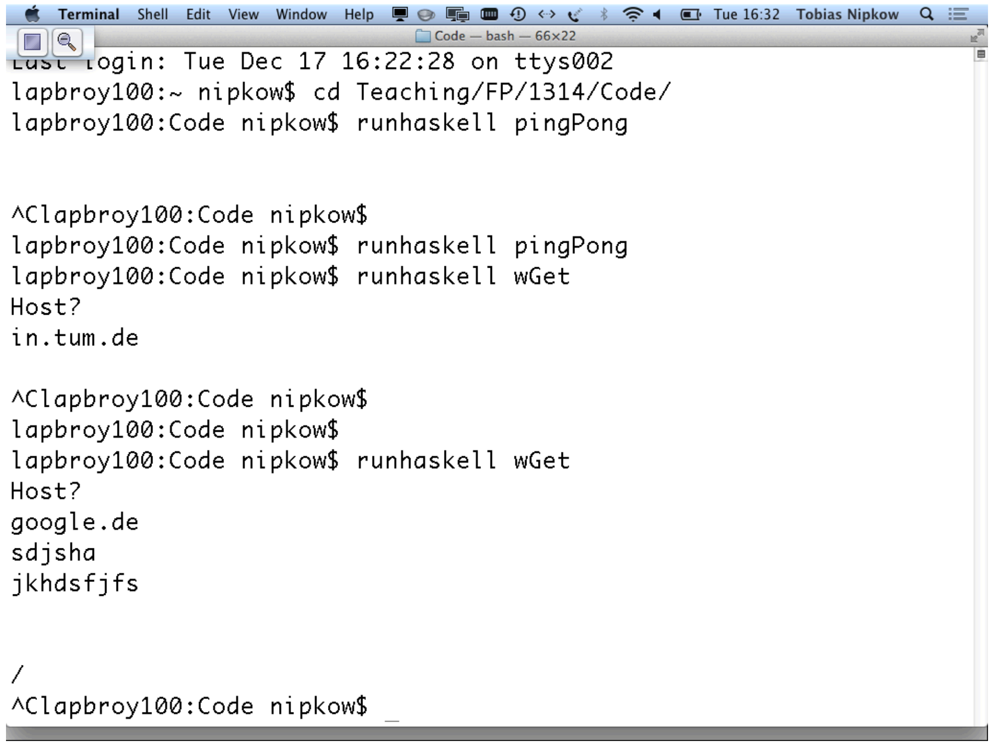
http://hackage.haskell.org/package/network/docs/Network-Socket.html



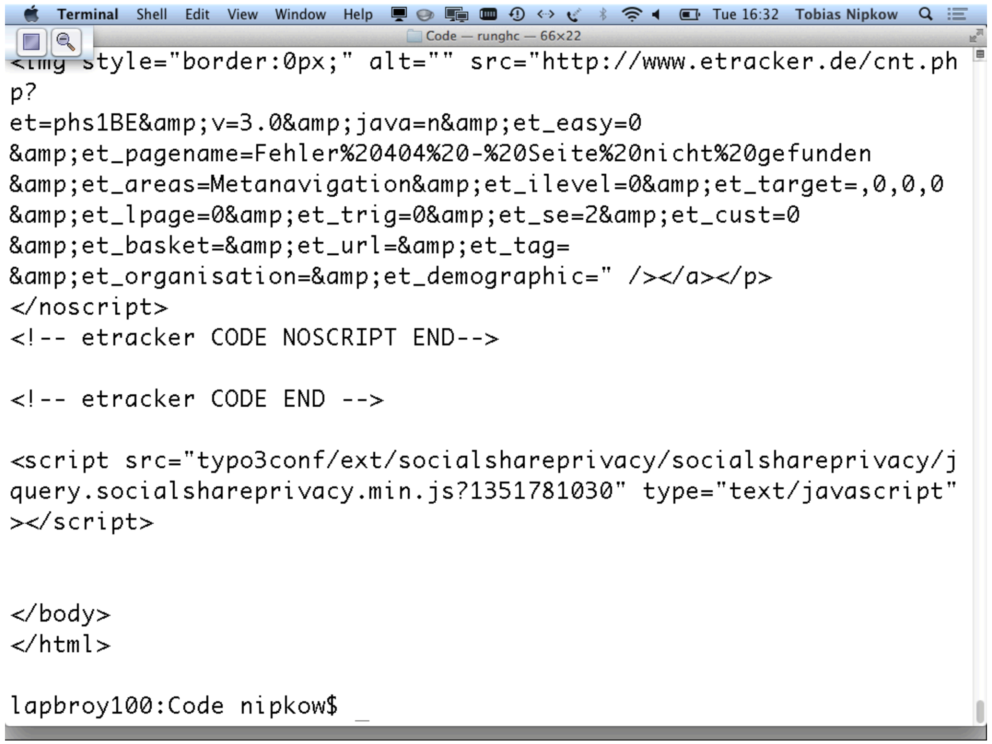
```
Last login: Tue Dec 17 16:22:28 on ttys002
lapbroy100:~ nipkow$ cd Teaching/FP/1314/Code/
lapbroy100:Code nipkow$ runhaskell pingPong


^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
lapbroy100:Code nipkow$ runhaskell wGet
Host?
in.tum.de

^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell wGet
Host?
```



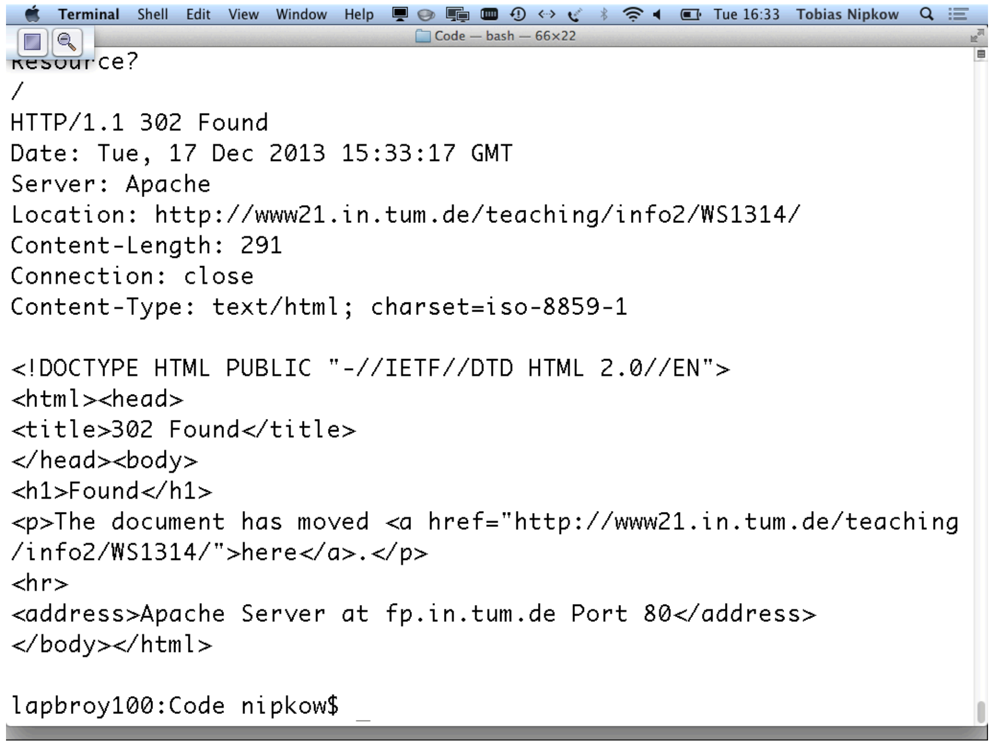
```
Last login: Tue Dec 17 16:22:28 on ttys002
lapbroy100:~ nipkow$ cd Teaching/FP/1314/Code/
lapbroy100:Code nipkow$ runhaskell pingPong


^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
lapbroy100:Code nipkow$ runhaskell wGet
Host?
in.tum.de

^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell wGet
Host?
google.de
sdjsha
jkhdsfjfs


/
^Clapbroy100:Code nipkow$ _
```



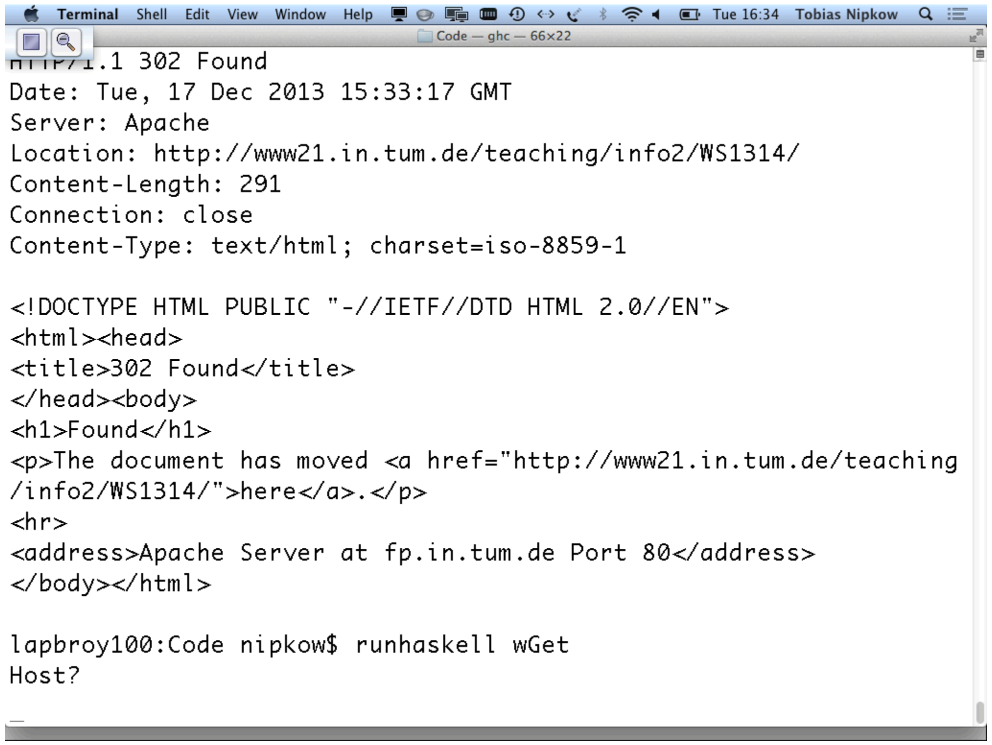
```
Last login: Tue Dec 17 16:22:28 on ttys002
lapbroy100:~ nipkow$ cd Teaching/FP/1314/Code/
lapbroy100:Code nipkow$ runhaskell pingPong


^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell pingPong
lapbroy100:Code nipkow$ runhaskell wGet
Host?
in.tum.de

^Clapbroy100:Code nipkow$
lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell wGet
Host?
google.de
sdjsha
jkhdsfjfs


/
^Clapbroy100:Code nipkow$ _
```



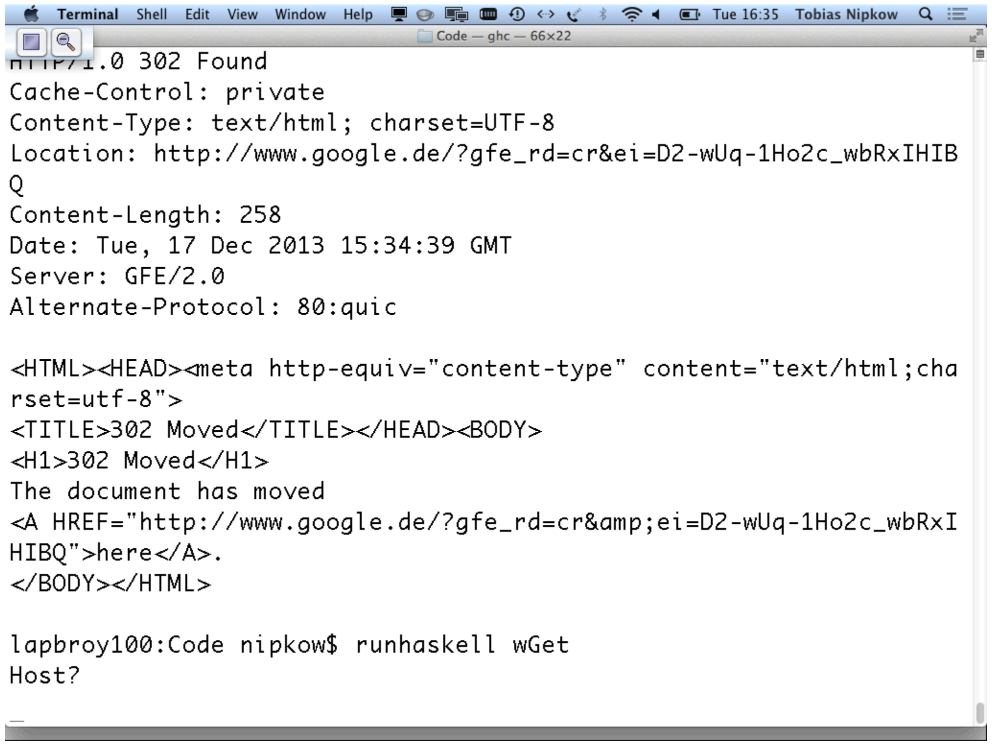
```
<img style="border:0px;" alt="" src="http://www.etracker.de/cnt.ph
p?
et=phs1BE&amp;v=3.0&amp;java=n&amp;et_easy=0
&amp;et_pagename=Fehler%20404%20-%20Seite%20nicht%20gefunden
&amp;et_areas=Metanavigation&amp;et_ilevel=0&amp;et_target=,0,0,0
&amp;et_lpage=0&amp;et_trig=0&amp;et_se=2&amp;et_cust=0
&amp;et_basket=&amp;et_url=&amp;et_tag=
&amp;et_organisation=&amp;et_demographic=" /></a></p>
</noscript>
<!-- etracker CODE NOSCRIPT END-->

<!-- etracker CODE END -->

<script src="typo3conf/ext/socialshareprivacy/socialshareprivacy/j
query.socialshareprivacy.min.js?1351781030" type="text/javascript"
></script>



</body>
</html>

lapbroy100:Code nipkow$ _
```



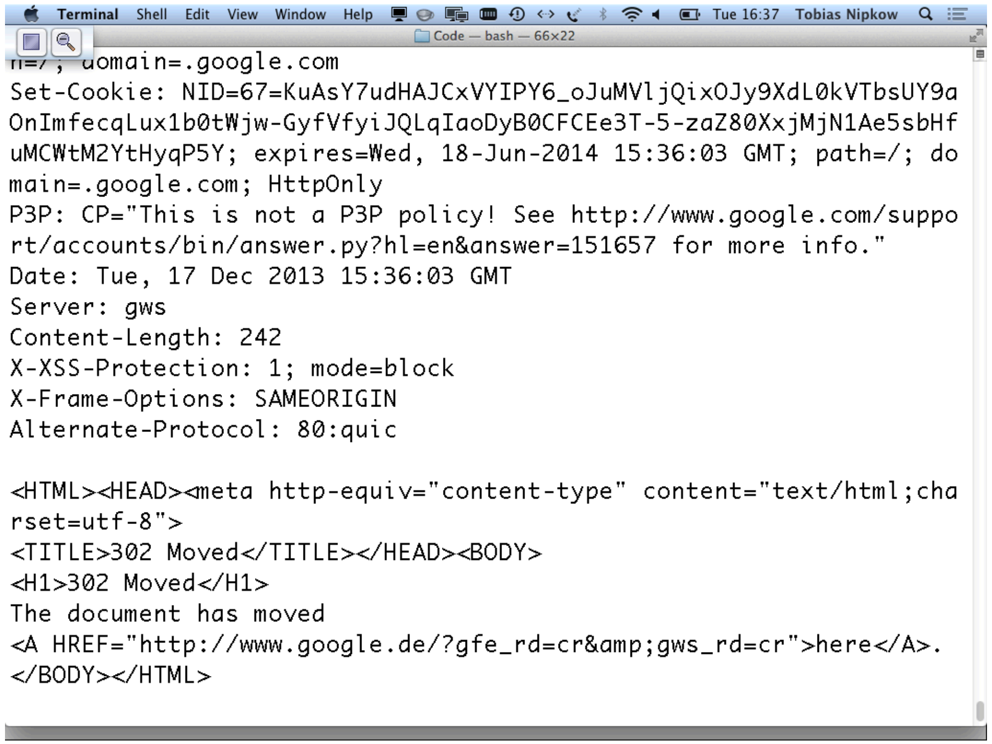
```
Resource?
/
HTTP/1.1 302 Found
Date: Tue, 17 Dec 2013 15:33:17 GMT
Server: Apache
Location: http://www21.in.tum.de/teaching/info2/WS1314/
Content-Length: 291
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www21.in.tum.de/teaching
/info2/WS1314/">here</a>.</p>
<hr>
<address>Apache Server at fp.in.tum.de Port 80</address>
</body></html>

lapbroy100:Code nipkow$ _
```


Terminal Shell Edit View Window Help Tue 16:34 Tobias Nipkow
Code — ghc — 66×22

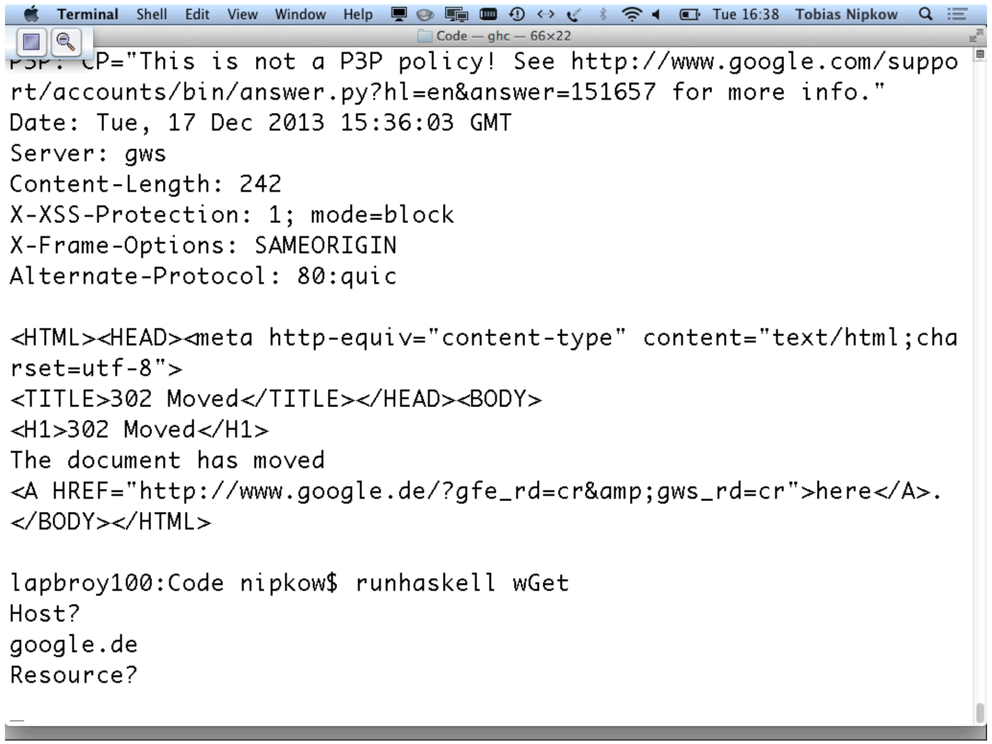
```
HTTP/1.1 302 Found
Date: Tue, 17 Dec 2013 15:33:17 GMT
Server: Apache
Location: http://www21.in.tum.de/teaching/info2/WS1314/
Content-Length: 291
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>302 Found</title>
</head><body>
<h1>Found</h1>
<p>The document has moved <a href="http://www21.in.tum.de/teaching
/info2/WS1314/">here</a>.</p>
<hr>
<address>Apache Server at fp.in.tum.de Port 80</address>
</body></html>

lapbroy100:Code nipkow$ runhaskell wGet
Host?
```


Terminal Shell Edit View Window Help Tue 16:35 Tobias Nipkow
Code — ghc — 66×22

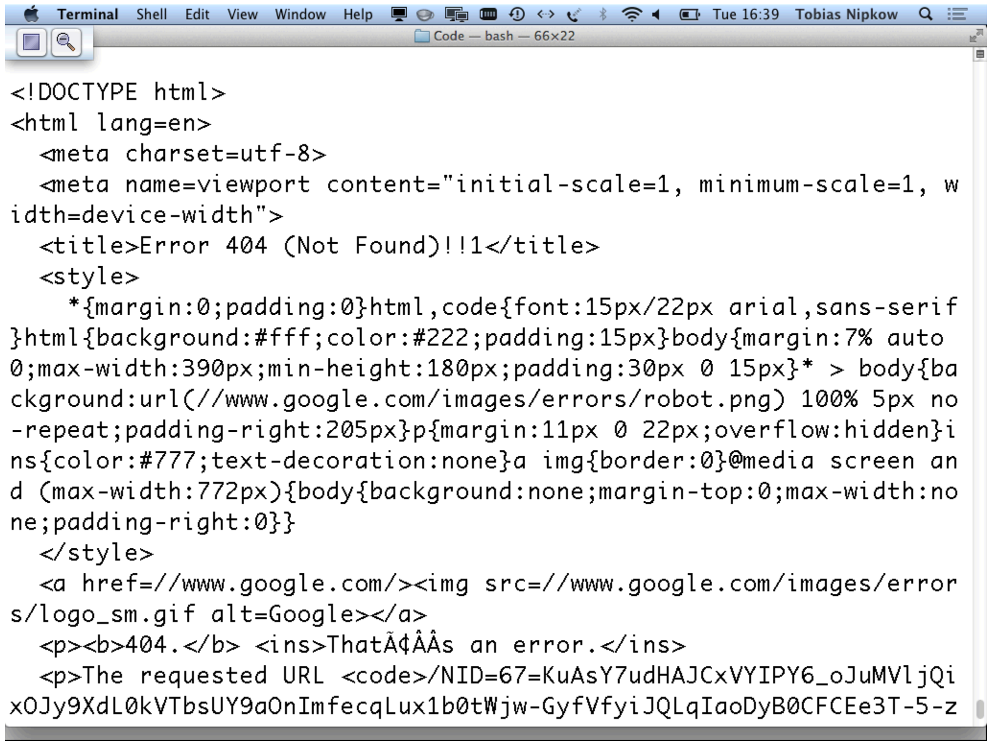
```
HTTP/1.0 302 Found
Cache-Control: private
Content-Type: text/html; charset=UTF-8
Location: http://www.google.de/?gfe_rd=cr&ei=D2-wUq-1Ho2c_wbRxIHIB
Q
Content-Length: 258
Date: Tue, 17 Dec 2013 15:34:39 GMT
Server: GFE/2.0
Alternate-Protocol: 80:quic

<HTML><HEAD><meta http-equiv="content-type" content="text/html;cha
rset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.de/?gfe_rd=cr&amp;ei=D2-wUq-1Ho2c_wbRxI
HIBQ">here</A>.
</BODY></HTML>

lapbroy100:Code nipkow$ runhaskell wGet
Host?
```


Terminal Shell Edit View Window Help Tue 16:37 Tobias Nipkow
Code — bash — 66×22

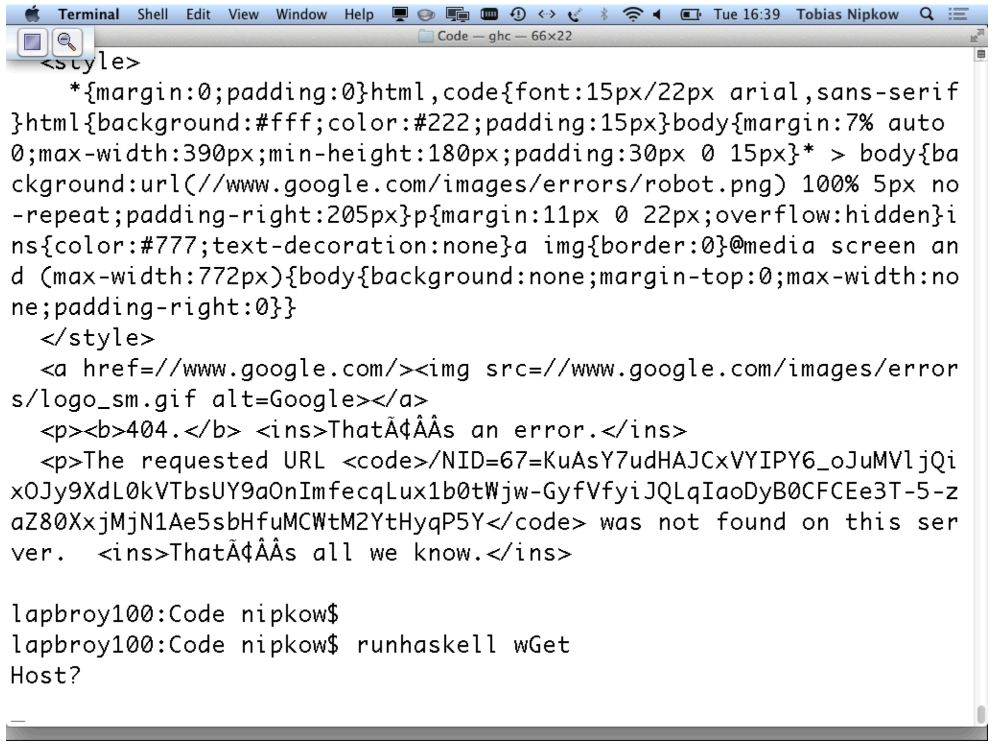
```
h=/; domain=.google.com
Set-Cookie: NID=67=KuAsY7udHAJCxVYIPY6_oJuMVljQixOJy9XdL0kVTbsUY9a
OnImfecqLux1b0tWjw-GyfVfyiJQLqIaoDyB0CFCEe3T-5-zaZ80XxjMjN1Ae5sbHf
uMCWtM2YtHyqP5Y; expires=Wed, 18-Jun-2014 15:36:03 GMT; path=/; do
main=.google.com; HttpOnly
P3P: CP="This is not a P3P policy! See http://www.google.com/suppo
rt/accounts/bin/answer.py?hl=en&answer=151657 for more info."
Date: Tue, 17 Dec 2013 15:36:03 GMT
Server: gws
Content-Length: 242
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic

<HTML><HEAD><meta http-equiv="content-type" content="text/html;cha
rset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.de/?gfe_rd=cr&amp;gws_rd=cr">here</A>.
</BODY></HTML>
```


Terminal Shell Edit View Window Help Tue 16:38 Tobias Nipkow
Code — ghc — 66×22

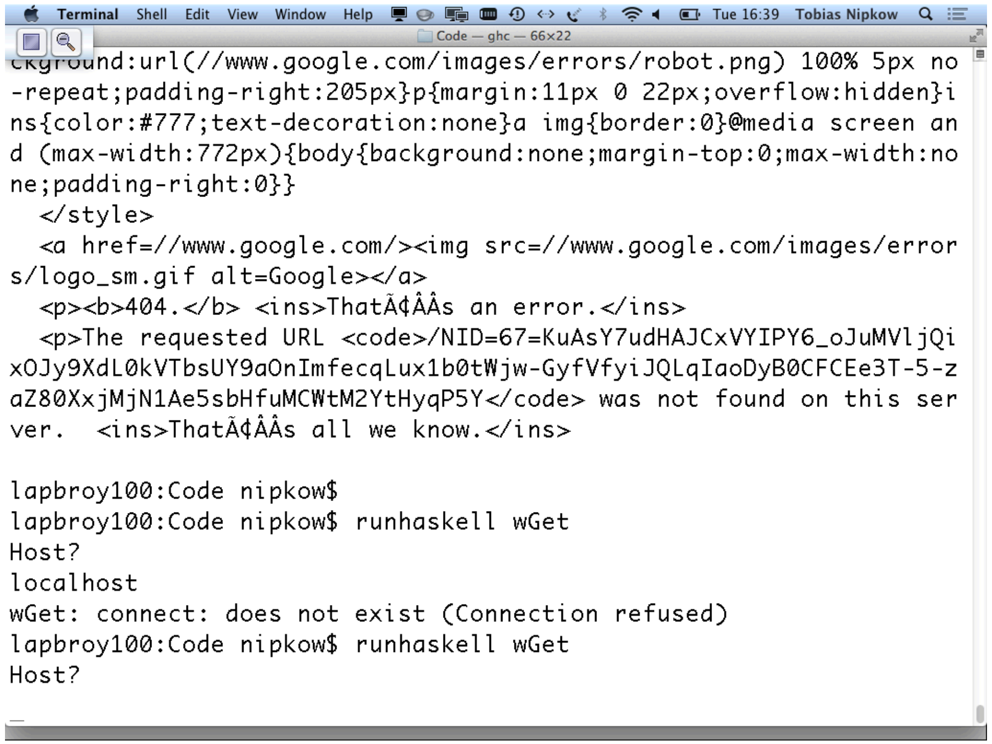
```
P3P: CP="This is not a P3P policy! See http://www.google.com/suppo
rt/accounts/bin/answer.py?hl=en&answer=151657 for more info."
Date: Tue, 17 Dec 2013 15:36:03 GMT
Server: gws
Content-Length: 242
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Alternate-Protocol: 80:quic

<HTML><HEAD><meta http-equiv="content-type" content="text/html;cha
rset=utf-8">
<TITLE>302 Moved</TITLE></HEAD><BODY>
<H1>302 Moved</H1>
The document has moved
<A HREF="http://www.google.de/?gfe_rd=cr&amp;gws_rd=cr">here</A>.
</BODY></HTML>

lapbroy100:Code nipkow$ runhaskell wGet
Host?
google.de
Resource?
```


Terminal Shell Edit View Window Help Tue 16:39 Tobias Nipkow
Code — bash — 66×22

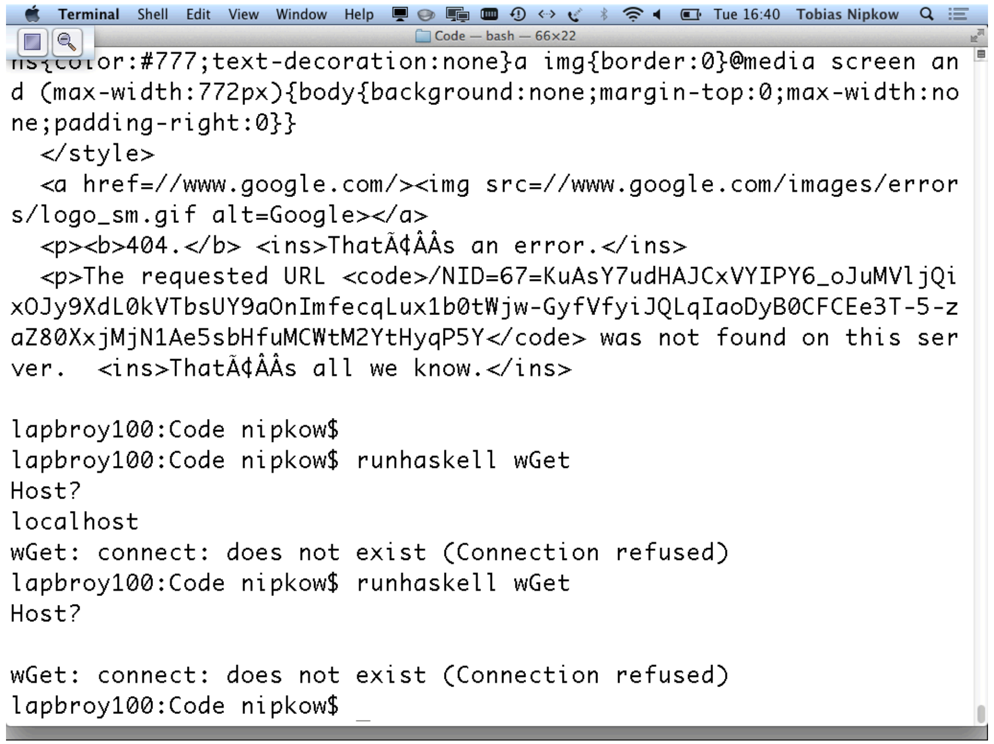
```
<!DOCTYPE html>
<html lang=en>
  <meta charset=utf-8>
  <meta name=viewport content="initial-scale=1, minimum-scale=1, w
idth=device-width">
  <title>Error 404 (Not Found)!!1</title>
  <style>
    *{margin:0;padding:0}html,code{font:15px/22px arial,sans-serif
}html{background:#fff;color:#222;padding:15px}body{margin:7% auto
0;max-width:390px;min-height:180px;padding:30px 0 15px}* > body{ba
ckground:url(//www.google.com/images/errors/robot.png) 100% 5px no
-repeat;padding-right:205px}p{margin:11px 0 22px;overflow:hidden}i
ns{color:#777;text-decoration:none}a img{border:0}@media screen an
d (max-width:772px){body{background:none;margin-top:0;max-width:no
ne;padding-right:0}}
  </style>
  <a href=//www.google.com/><img src=//www.google.com/images/error
s/logo_sm.gif alt=Google></a>
  <p><b>404.</b> <ins>ThatÃ¢ÂÂs an error.</ins>
  <p>The requested URL <code>/NID=67=KuAsY7udHAJCxVYIPY6_oJuMVljQi
xOJy9XdL0kVTbsUY9aOnImfecqLux1b0tWjw-GyfVfyiJQLqIaoDyB0CFCEe3T-5-z
```


Terminal Shell Edit View Window Help Tue 16:39 Tobias Nipkow
Code — ghc — 66×22

```
  <style>
    *{margin:0;padding:0}html,code{font:15px/22px arial,sans-serif
}html{background:#fff;color:#222;padding:15px}body{margin:7% auto
0;max-width:390px;min-height:180px;padding:30px 0 15px}* > body{ba
ckground:url(//www.google.com/images/errors/robot.png) 100% 5px no
-repeat;padding-right:205px}p{margin:11px 0 22px;overflow:hidden}i
ns{color:#777;text-decoration:none}a img{border:0}@media screen an
d (max-width:772px){body{background:none;margin-top:0;max-width:no
ne;padding-right:0}}
  </style>
  <a href=//www.google.com/><img src=//www.google.com/images/error
s/logo_sm.gif alt=Google></a>
  <p><b>404.</b> <ins>ThatÃ¢ÂÂs an error.</ins>
  <p>The requested URL <code>/NID=67=KuAsY7udHAJCxVYIPY6_oJuMVljQi
xOJy9XdL0kVTbsUY9aOnImfecqLux1b0tWjw-GyfVfyiJQLqIaoDyB0CFCEe3T-5-z
aZ80XxjMjN1Ae5sbHfuMCWtM2YtHyqP5Y</code> was not found on this ser
ver.  <ins>ThatÃ¢ÂÂs all we know.</ins>

lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell wGet
Host?
_
```


Terminal Shell Edit View Window Help Tue 16:39 Tobias Nipkow
Code — ghc — 66×22

```
ckground:url(//www.google.com/images/errors/robot.png) 100% 5px no
-repeat;padding-right:205px}p{margin:11px 0 22px;overflow:hidden}i
ns{color:#777;text-decoration:none}a img{border:0}@media screen an
d (max-width:772px){body{background:none;margin-top:0;max-width:no
ne;padding-right:0}}
  </style>
  <a href=//www.google.com/><img src=//www.google.com/images/error
s/logo_sm.gif alt=Google></a>
  <p><b>404.</b> <ins>ThatÃ¢ÂÂs an error.</ins>
  <p>The requested URL <code>/NID=67=KuAsY7udHAJCxVYIPY6_oJuMVljQi
xOJy9XdL0kVTbsUY9aOnImfecqLux1b0tWjw-GyfVfyiJQLqIaoDyB0CFCEe3T-5-z
aZ80XxjMjN1Ae5sbHfuMCWtM2YtHyqP5Y</code> was not found on this ser
ver.  <ins>ThatÃ¢ÂÂs all we know.</ins>

lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell wGet
Host?
localhost
wGet: connect: does not exist (Connection refused)
lapbroy100:Code nipkow$ runhaskell wGet
Host?
_
```


Terminal Shell Edit View Window Help Tue 16:40 Tobias Nipkow
Code — bash — 66×22

```
ns{color:#777;text-decoration:none}a img{border:0}@media screen an
d (max-width:772px){body{background:none;margin-top:0;max-width:no
ne;padding-right:0}}
  </style>
  <a href=//www.google.com/><img src=//www.google.com/images/error
s/logo_sm.gif alt=Google></a>
  <p><b>404.</b> <ins>ThatÃ¢ÂÂs an error.</ins>
  <p>The requested URL <code>/NID=67=KuAsY7udHAJCxVYIPY6_oJuMVljQi
xOJy9XdL0kVTbsUY9aOnImfecqLux1b0tWjw-GyfVfyiJQLqIaoDyB0CFCEe3T-5-z
aZ80XxjMjN1Ae5sbHfuMCWtM2YtHyqP5Y</code> was not found on this ser
ver.  <ins>ThatÃ¢ÂÂs all we know.</ins>

lapbroy100:Code nipkow$
lapbroy100:Code nipkow$ runhaskell wGet
Host?
localhost
wGet: connect: does not exist (Connection refused)
lapbroy100:Code nipkow$ runhaskell wGet
Host?

wGet: connect: does not exist (Connection refused)
lapbroy100:Code nipkow$ _
```