

# 1. Computers

Prof. Dr. Charles Wüthrich  
B.Sc. Francesco Andreussi,  
CoGVis/MMC, Faculty of Media  
Bauhaus-University Weimar

# A Computer

- From a young age, today people are used to use one or more computers



# A Computer

- From a young age, today people are used to use one or more computers



# A Computer

- From a young age, today people are used to use one or more computers



Courtesy Bundesarchiv

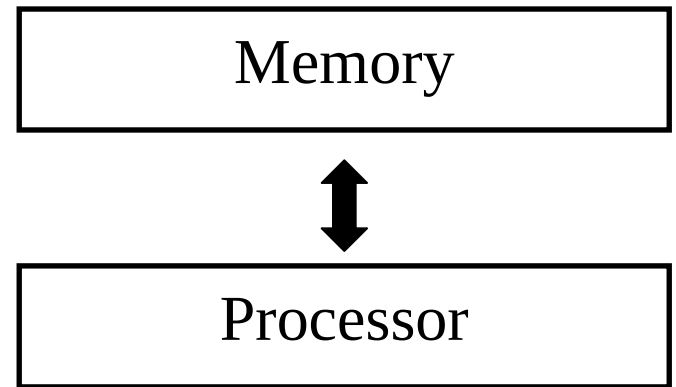
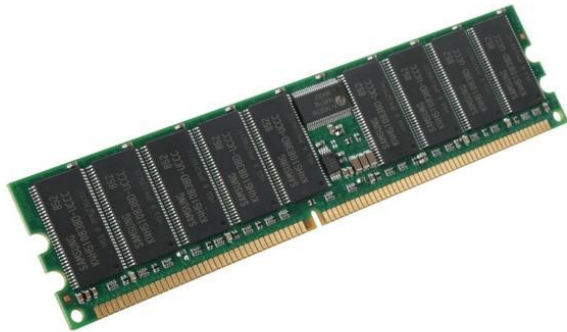
# What makes a computer?

# A minimal computer

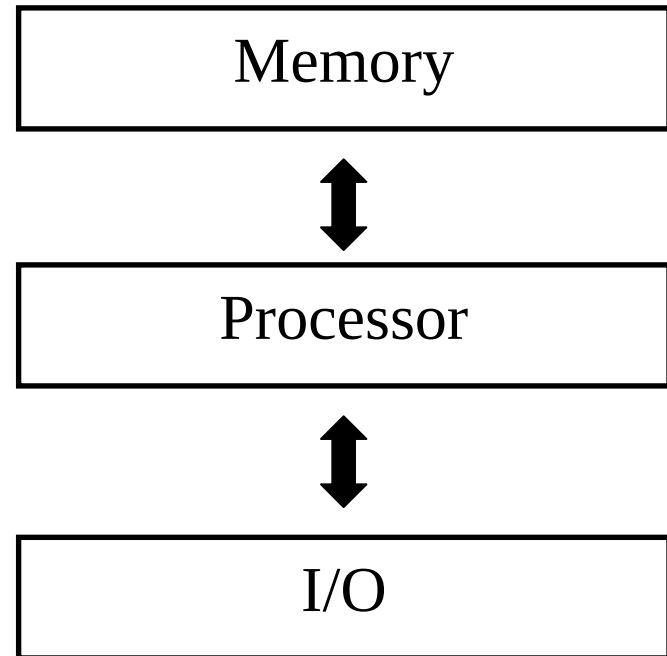
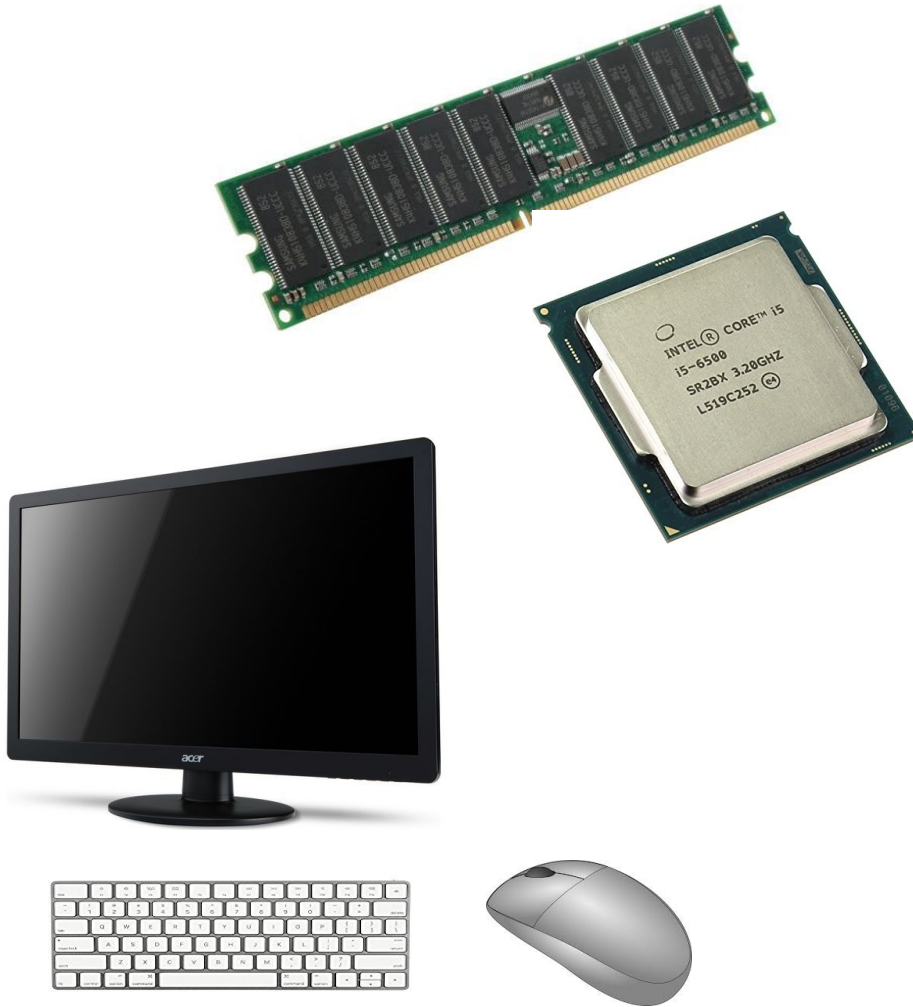


Processor

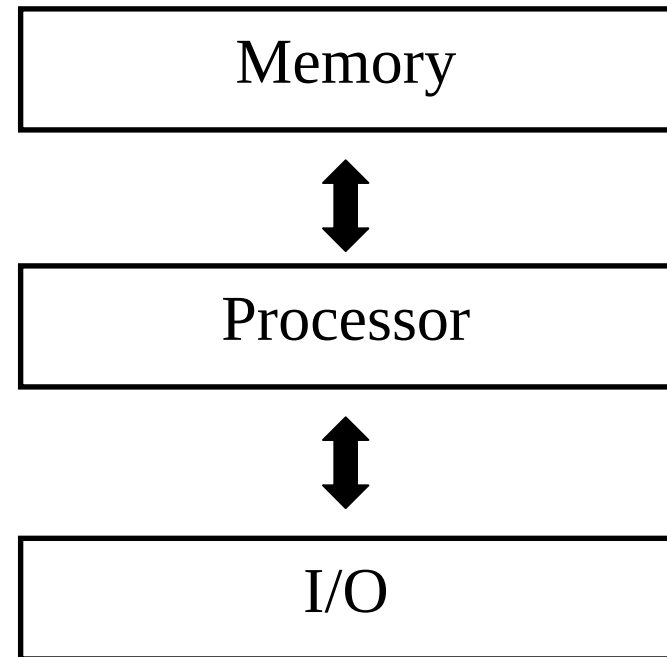
# A minimal computer



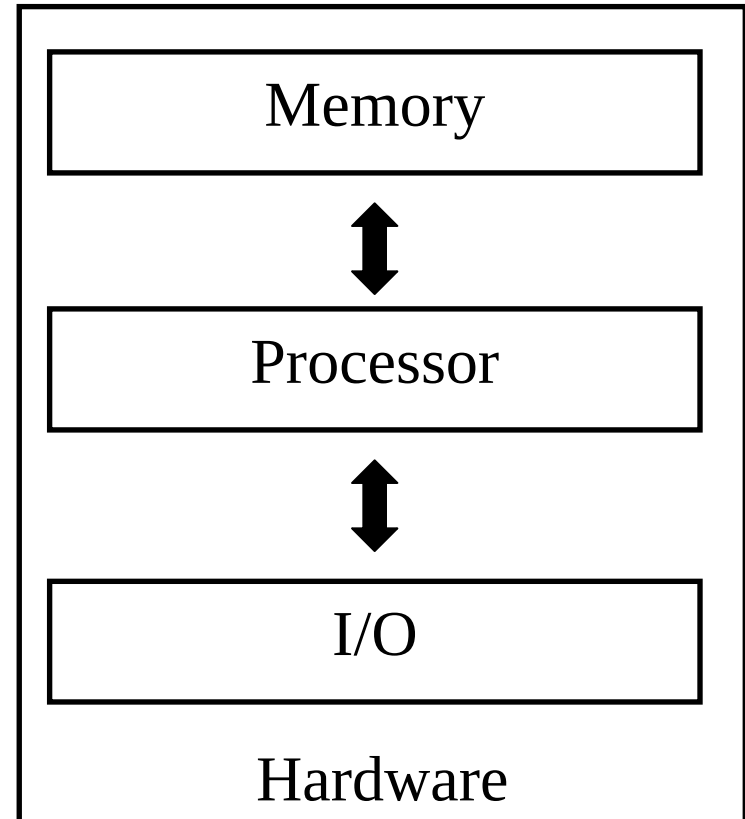
# A minimal computer



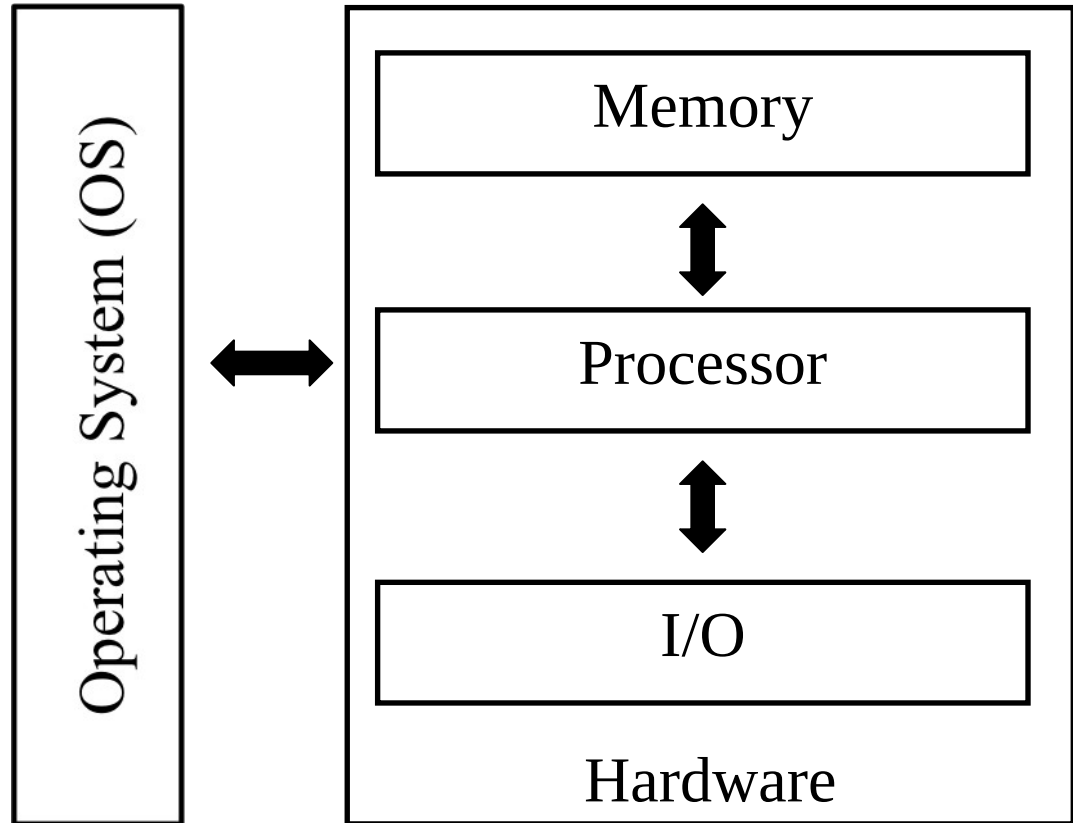
# A minimal computer



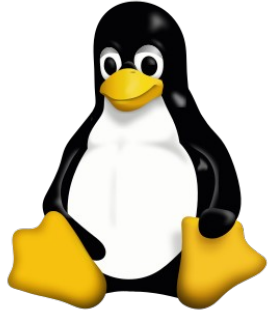
# A minimal computer



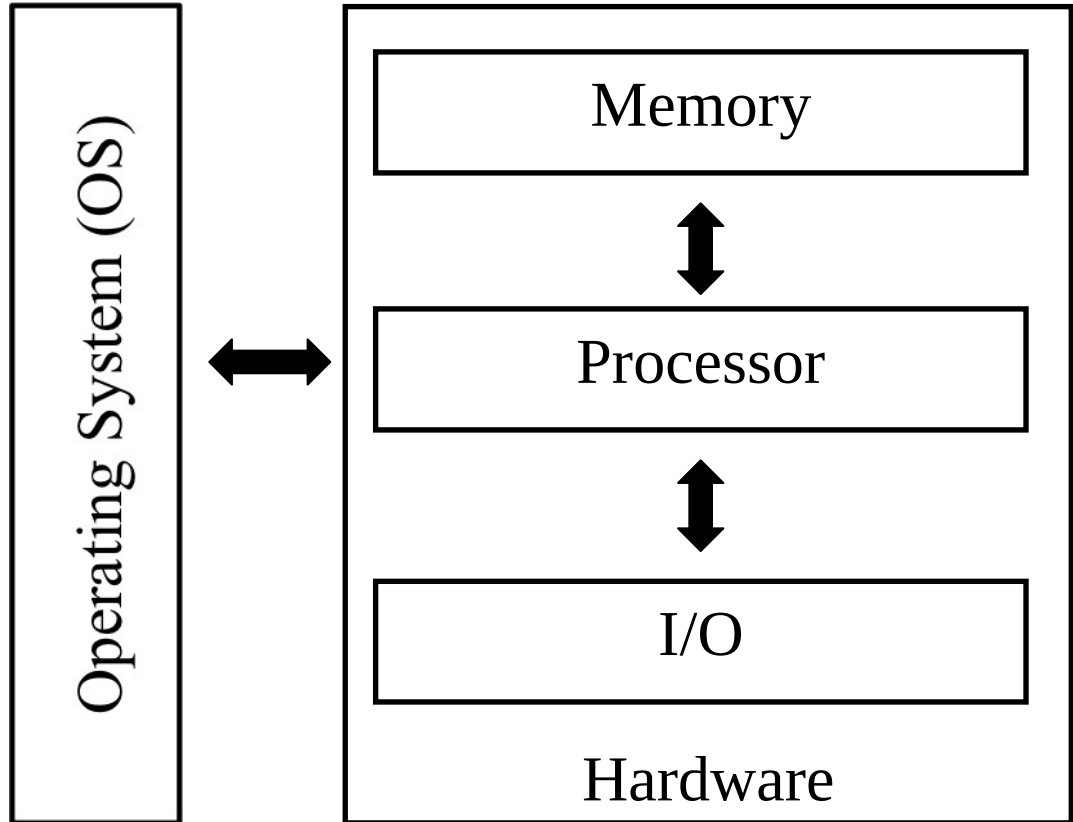
# A less minimal computer



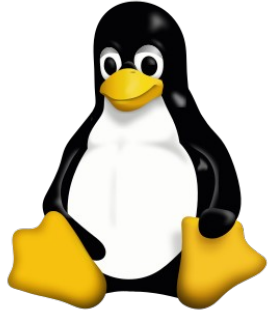
# A less minimal computer



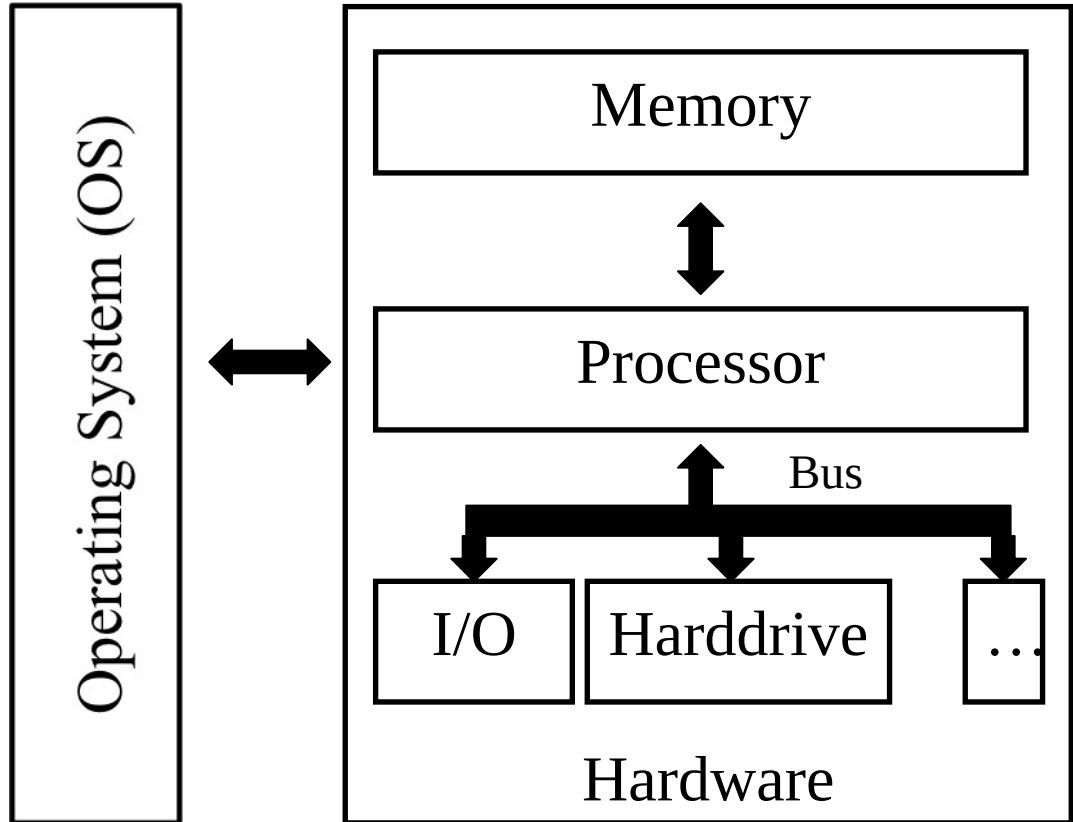
Mac OS



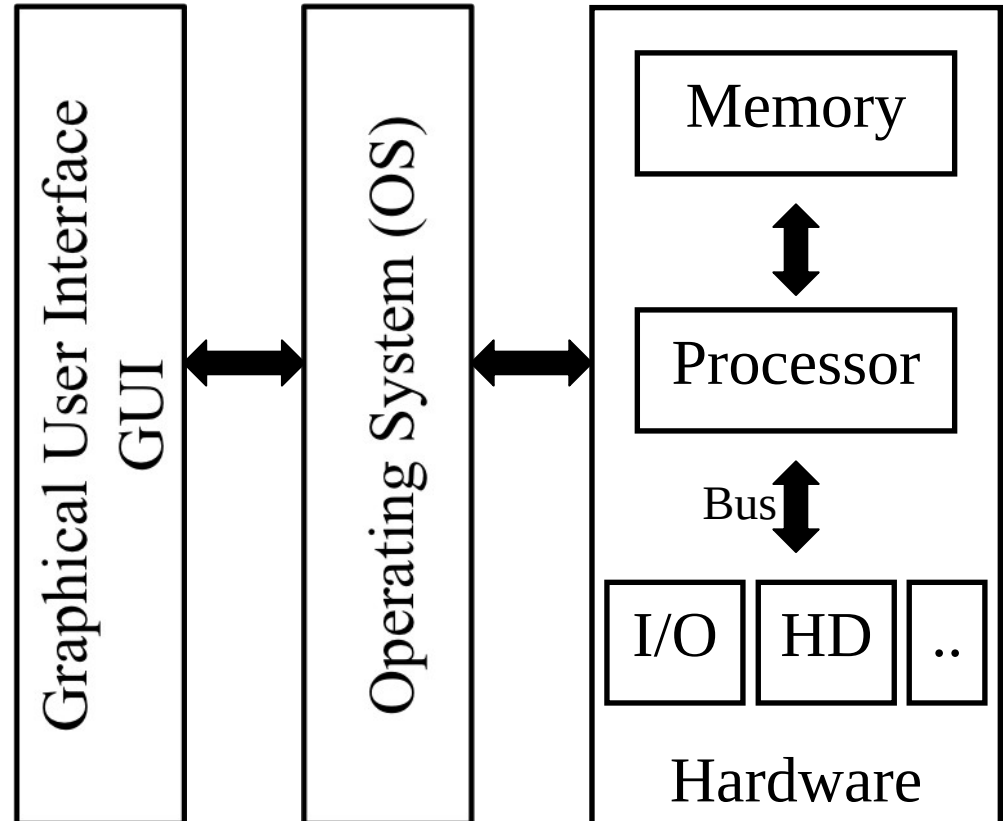
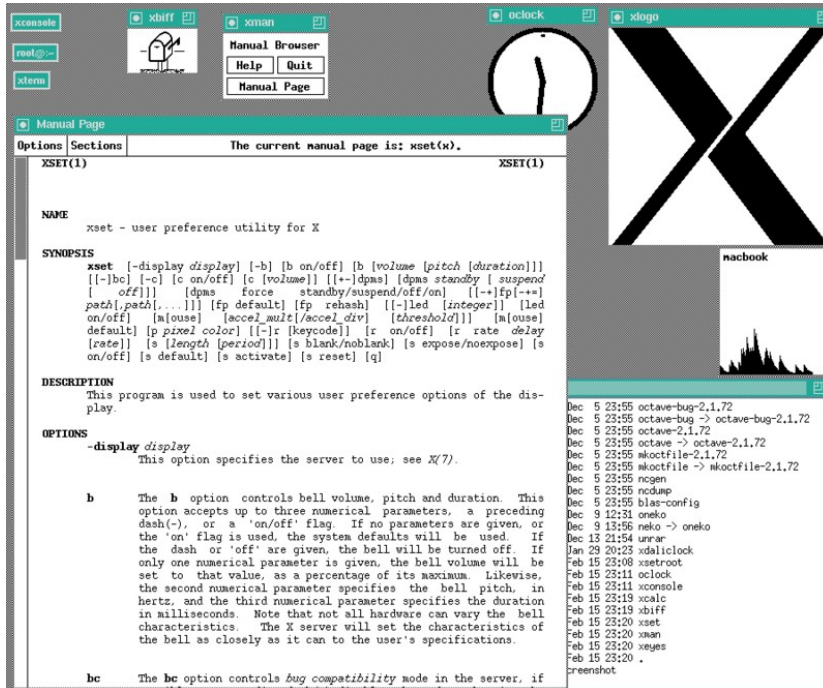
# A more realistic computer



Mac OS



# A not quite so minimal computer



# A not quite so minimal computer

The screenshot displays a windowed desktop environment with several open windows:

- xconsole**: Shows a terminal prompt `root@:-`.
- xterm**: A terminal window.
- xbiff**: A window with a drawing of a person's head.
- xman**: A "Manual Browser" window with buttons for "Help", "Quit", and "Manual Page".
- oclock**: A clock window showing a large '1'.
- xlogo**: A window displaying a large black 'X' logo.
- Manual Page**: The main window showing the manual for `xset(1)`. It includes sections for NAME, SYNOPSIS, DESCRIPTION, and OPTIONS.
- nacbook**: A window showing a waveform graph.
- Terminal**: A window showing system logs with timestamps and program names like `octave-bug-2.1.72`, `oneko`, `unrar`, `xdaliclock`, `xsetroot`, `oclock`, `xconsole`, `xcalc`, `xbiff`, `xset`, `xman`, `xeyes`, and `creenshot`.

**Manual Page Content:**

```
XSET(1) XSET(1)

NAME
  xset - user preference utility for X

SYNOPSIS
  xset [-display display] [-b] [b on/off] [b [volume [pitch [duration]]]
  [[-]bc] [-c] [c on/off] [c [volume]] [[+|-]dpms] [dpms standby | suspend
  [ off]]] [dpms force standby/suspend/off/on] [[+|fp[+|=]
  path[,path[...]]] [fp default] [fp rehash] [[-]led [integer]] [led
  on/off] [m[ouse] [accel_mult[/accel_div] [threshold]]] [m[ouse]
  default] [p pixel color] [[-]r [keycode]] [r on/off] [r rate delay
  [rate]] [s [length [period]]] [s blank/noblank] [s expose/noexpose] [s
  on/off] [s default] [s activate] [s reset] [q]

DESCRIPTION
  This program is used to set various user preference options of the display.

OPTIONS
  -display display
    This option specifies the server to use; see X(7).

  b
    The b option controls bell volume, pitch and duration. This
    option accepts up to three numerical parameters, a preceding
    dash(-), or a 'on/off' flag. If no parameters are given, or
    the 'on' flag is used, the system defaults will be used. If
    the dash or 'off' are given, the bell will be turned off. If
    only one numerical parameter is given, the bell volume will be
    set to that value, as a percentage of its maximum. Likewise,
    the second numerical parameter specifies the bell pitch, in
    hertz, and the third numerical parameter specifies the duration
    in milliseconds. Note that not all hardware can vary the bell
    characteristics. The X server will set the characteristics of
    the bell as closely as it can to the user's specifications.

  bc
    The bc option controls bug compatibility mode in the server, if
```

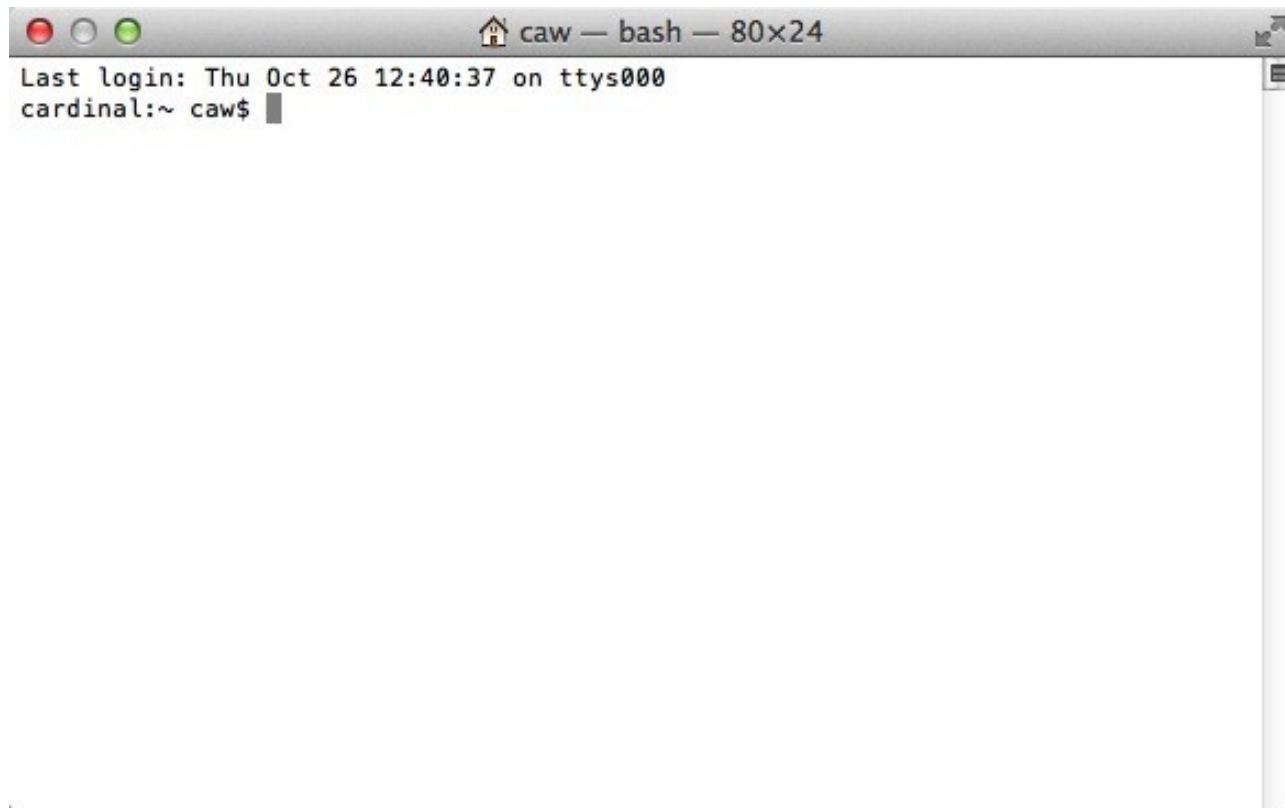
# The file system

- All data organized in a file system (on your HD) like a tree
- Root directory: `/`
- Subdirectories:
  - `/dir1/`
  - `/dir2/`
  - ...
- Subsubdirectories
  - `/dir1/sub1`
  - `/sub2`
  - `/...`
  - `/dir2/sub1`
  - `/sub2`
  - `/...`
  - ... and so on...

# Command Line Interface (CLI)

- All systems have a Command Line Interface where you can type commands for the computer

Mac: Terminal



The image shows a screenshot of a Mac Terminal window. The title bar at the top reads "caw — bash — 80x24". The terminal content displays the following text: "Last login: Thu Oct 26 12:40:37 on ttys000" followed by the prompt "cardinal:~ caw\$" with a cursor. The window has standard Mac OS window controls (red, yellow, green buttons) and a scroll bar on the right side.

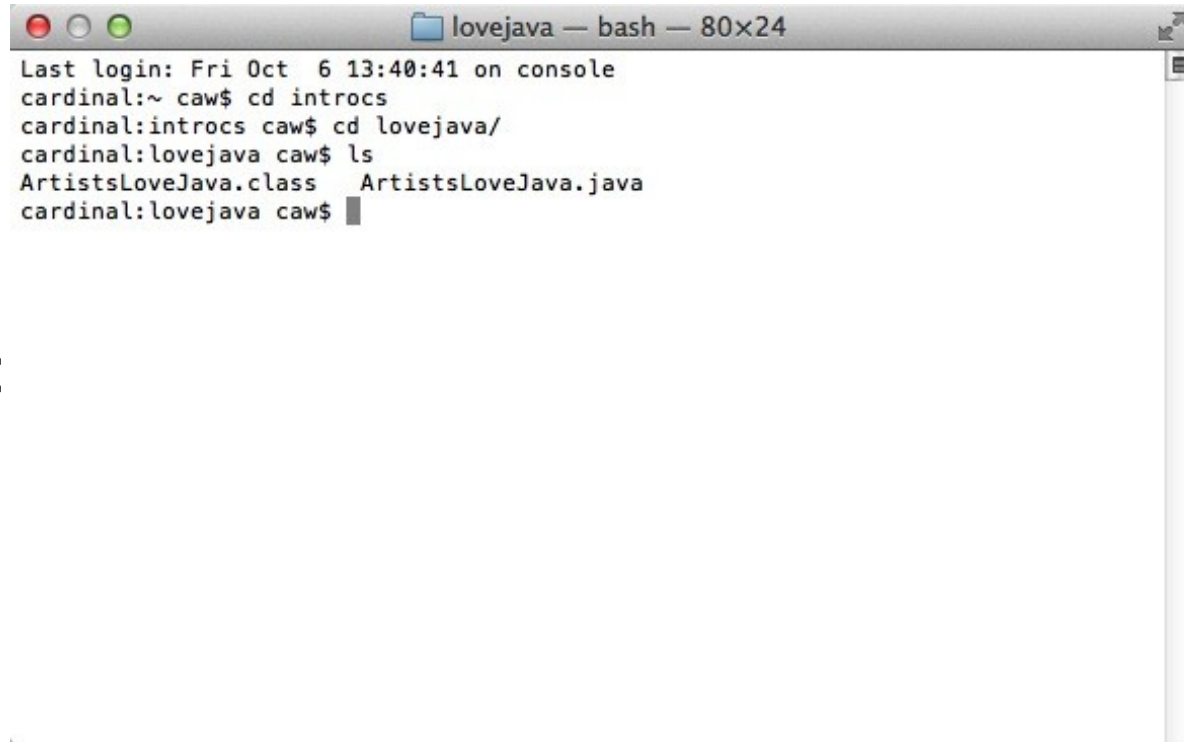
# Moving in the file system

- Change directory you are working at:  
from the CLI type

cd

- Example:

cd /

A terminal window titled "lovejava — bash — 80x24" showing a sequence of commands and their outputs. The user starts in the home directory (~), navigates to the "introsocs" directory, then to the "lovejava/" directory, and finally lists the contents of the "lovejava/" directory, which includes "ArtistsLoveJava.class" and "ArtistsLoveJava.java".

```
Last login: Fri Oct 6 13:40:41 on console
cardinal:~ caw$ cd introsocs
cardinal:introcs caw$ cd lovejava/
cardinal:lovejava caw$ ls
ArtistsLoveJava.class  ArtistsLoveJava.java
cardinal:lovejava caw$
```

- Current directory:

.

- Up directory

..

cd ..

# Listing files

- From any directory position:

```
ls
```

- Example:

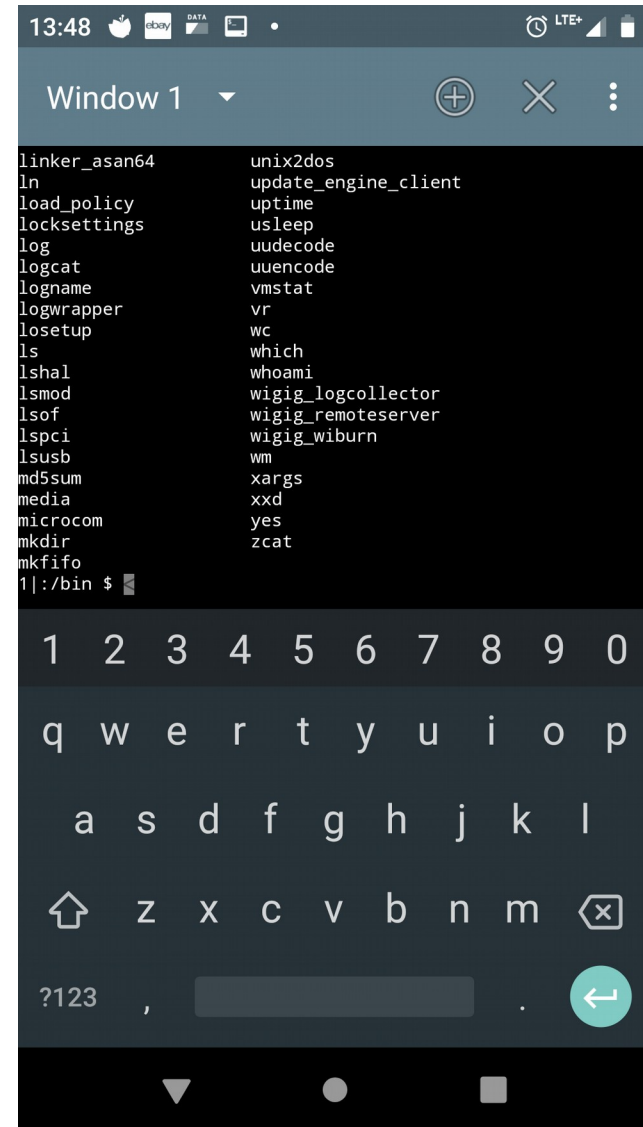
```
ls ./introscs/
```

```
ArtistsLoveJava.java      hello      lovejava
checkstyle-5.5            java3d     stdlib.jar
findbugs-2.0.3            log.txt
```

- Lists the subdirectory of the current directory named intros

# Listing files

- If you think “This guy is telling me bollocks” Think again!
- Download a “Terminal Emulator” from GooglePlay
- And take a look at what is under your pretty cellphone!!!!
- Absolutely the same!!!!



The screenshot shows a terminal emulator window titled "Window 1" on a smartphone. The terminal displays a list of system files and utilities, including linker\_asan64, ln, load\_policy, locksettings, log, logcat, logname, logwrapper, losetup, ls, lshal, lsmod, lsuf, lspci, lsusb, md5sum, media, microcom, mkdir, mkfifo, unix2dos, update\_engine\_client, uptime, usleep, uuencode, vmstat, vr, wc, which, whoami, wigig\_logcollector, wigig\_remoteserver, wigig\_wiburn, wm, xargs, xxd, yes, and zcat. The prompt is 1|:/bin \$.

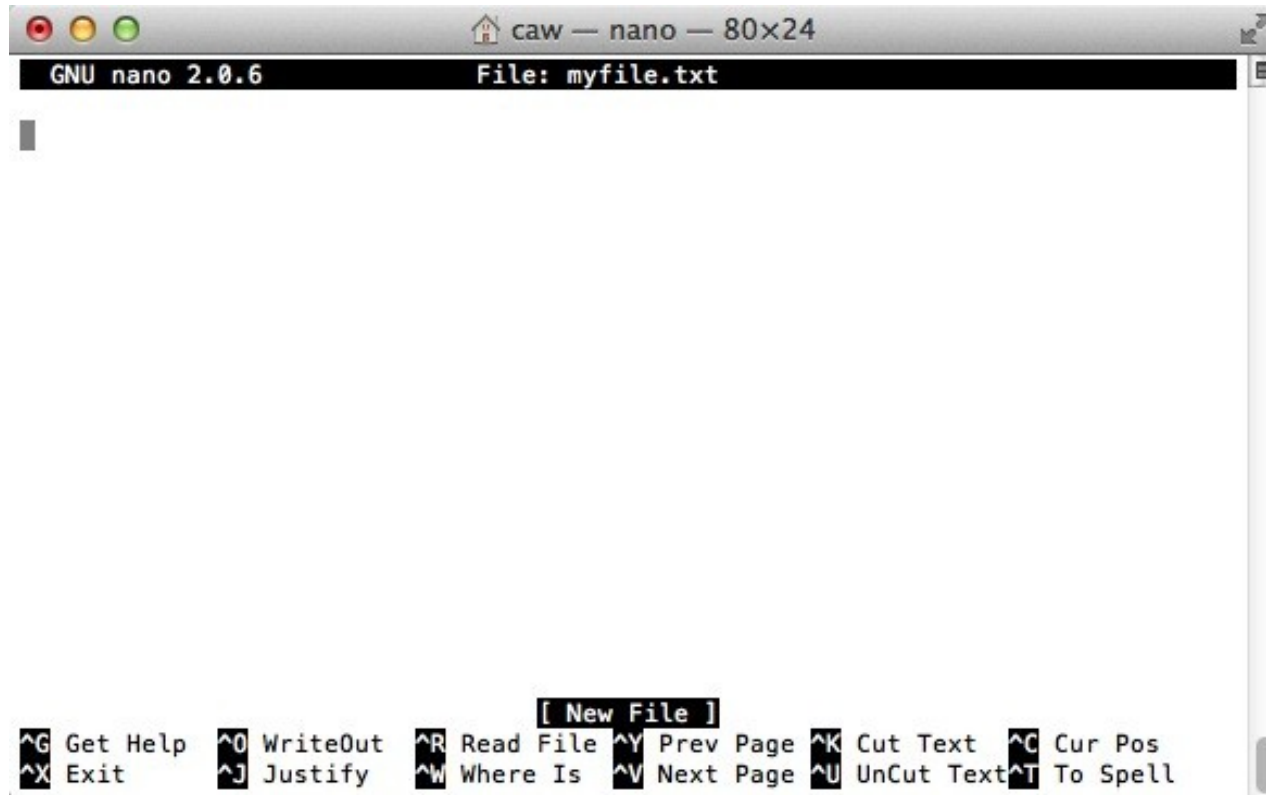
# Editing files

- You will need an editor application, such as:

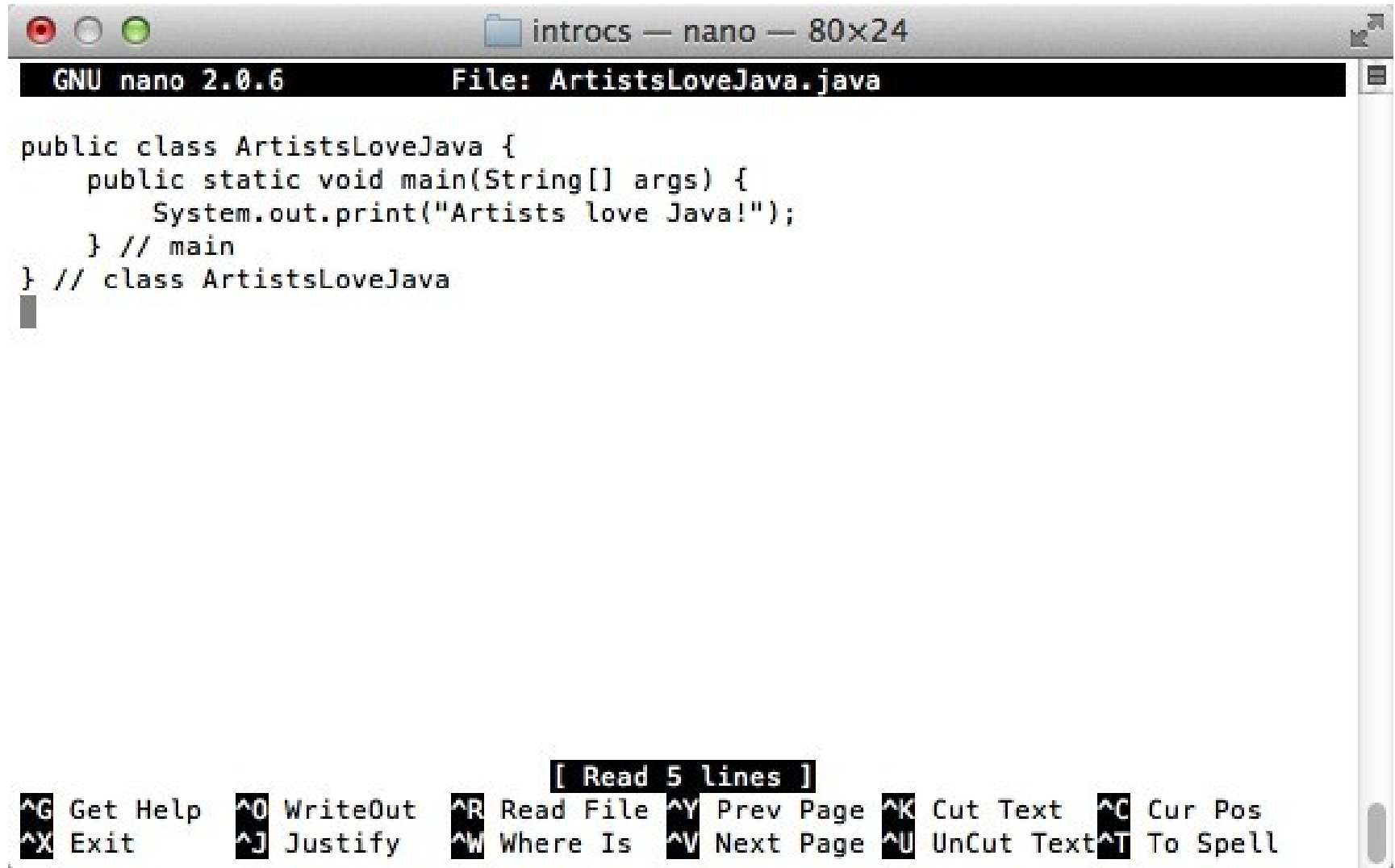
`nano emacs . . . .`

- Example:

`nano myfile.txt`



# My first program



The image shows a terminal window with a grey title bar that reads "intros — nano — 80x24". Below the title bar is a black header bar with "GNU nano 2.0.6" on the left and "File: ArtistsLoveJava.java" on the right. The main area of the terminal contains the following Java code:

```
public class ArtistsLoveJava {  
    public static void main(String[] args) {  
        System.out.print("Artists love Java!");  
    } // main  
} // class ArtistsLoveJava
```

At the bottom of the terminal, there is a status bar with the text "[ Read 5 lines ]" in a black box. Below this, there are two rows of keyboard shortcuts:

<b>^G</b> Get Help	<b>^O</b> WriteOut	<b>^R</b> Read File	<b>^Y</b> Prev Page	<b>^K</b> Cut Text	<b>^C</b> Cur Pos
<b>^X</b> Exit	<b>^J</b> Justify	<b>^W</b> Where Is	<b>^V</b> Next Page	<b>^U</b> UnCut Text	<b>^T</b> To Spell

# Compiling my first program

```
lovejava — bash — 80x24
Last login: Fri Oct 6 16:16:23 on ttys000
cardinal:~ caw$ cd introcs
cardinal:introcs caw$ cd lovejava
cardinal:lovejava caw$ ls
ArtistsLoveJava.class ArtistsLoveJava.java
cardinal:lovejava caw$ more ArtistsLoveJava.java
public class ArtistsLoveJava {
    public static void main(String[] args) {
        System.out.print("Artists love Java!\n");
    } // main
} // class ArtistsLoveJava
cardinal:lovejava caw$ javac ArtistsLoveJava.java
cardinal:lovejava caw$ ls -al
total 16
drwxr-xr-x  4 caw  staff  136 Jul  7 16:05 .
drwxr-xr-x 11 caw  staff  374 Jun 28 18:22 ..
-rw-r--r--  1 caw  staff  441 Oct  6 16:41 ArtistsLoveJava.class
-rw-r--r--  1 caw  staff  173 Jul  7 16:05 ArtistsLoveJava.java
cardinal:lovejava caw$
```

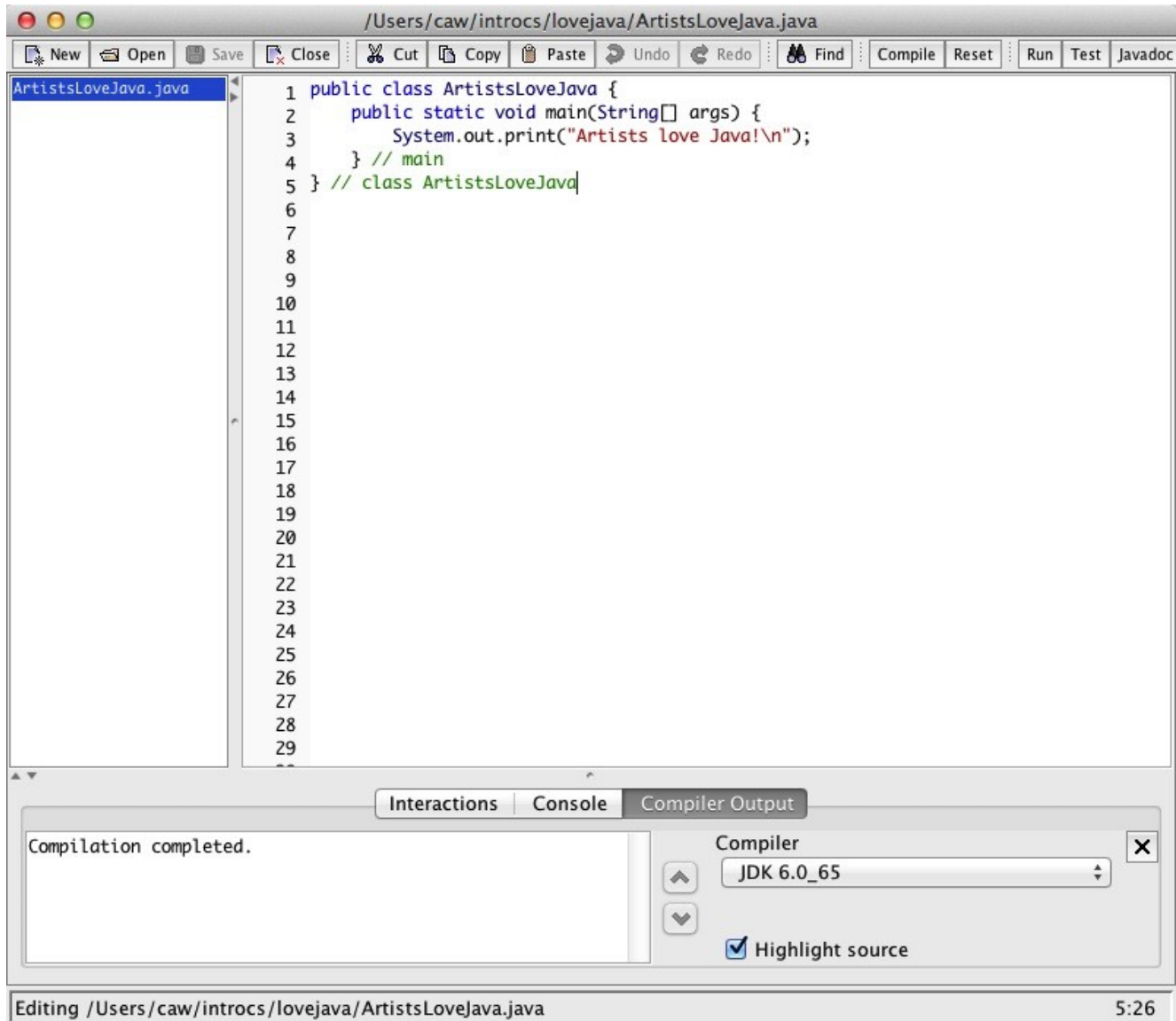
# Compiling my first program

- Compiling: a “translator” compiles your high level code into a low level code understandable by the machine
- A compiler does exactly this, and achieves machine (or in our case java virtual machine) executable code
- This code is then ready to be “executed” by the machine so as to “run” your instructions

# Running my first program

```
lovejava — bash — 80x24
Last login: Fri Oct 6 17:50:51 on ttys003
cardinal:~ caw$ cd introcs
cardinal:introcs caw$ cd lovejava
cardinal:lovejava caw$ ls
ArtistsLoveJava.class  ArtistsLoveJava.java
cardinal:lovejava caw$ more ArtistsLoveJava.java
public class ArtistsLoveJava {
    public static void main(String[] args) {
        System.out.print("Artists love Java!\n");
    } // main
} // class ArtistsLoveJava
cardinal:lovejava caw$ javac ArtistsLoveJava.java
cardinal:lovejava caw$ ls -al
total 16
drwxr-xr-x  4 caw  staff  136 Oct  6 17:47 .
drwxr-xr-x 11 caw  staff  374 Jun 28 18:22 ..
-rw-r--r--  1 caw  staff  441 Oct  6 17:55 ArtistsLoveJava.class
-rw-r--r--  1 caw  staff  173 Jul  7 16:05 ArtistsLoveJava.java
cardinal:lovejava caw$ java ArtistsLoveJava
Artists love Java!
cardinal:lovejava caw$
```

# My first program (IDE)



# My first program

---

```
public class ArtistsLoveJava {  
    public static void main(String[] args) {  
        System.out.print("Artists love Java!\n");  
    } // main  
} // class ArtistsLoveJava
```

---

# Errors

- When you write a program there are three types of errors that can happen
  - Compile time: the compiler cannot understand the code you typed in because it does not conform to the syntax
  - Execution time: the machine tries to execute the code you wrote (which compiles correctly).
    - Possible causes: Division by zero, trying to touch things that you are not allowed to touch (in the system)
  - Logical errors: most difficult to catch.  
Wrong results, program behaves erratically....

Catching these is both frustrating and rewarding.

# Controlling (a bit) Input and Output

- I'd like to discuss with you a new simple program:

---

```
public class ReadArgument {
    public static void main(String[] args) {
        System.out.print("My favourite pets are ");
        System.out.print(args[0]);
        System.out.println(". And yours?");
    } // main
} // ReadArgument
```

---

- This time, the program has more lines
- The computer starts at the *main* line, and executes each of the 3 lines one after the other
- What do these lines do?

# Controlling (a bit) Input and Output

---

```
public class ReadArgument {  
    public static void main(String[] args) {  
        System.out.print("My favourite pets are ");  
        System.out.print(args[0]);  
        System.out.println(". And yours?");  
    } // main  
} // ReadArgument
```

---

- The first line simply prints the content of the brackets in the CLI

# Controlling (a bit) Input and Output

---

```
public class ReadArgument {  
    public static void main(String[] args) {  
        System.out.print("My favourite pets are ");  
        System.out.print(args[0]);  
        System.out.println(". And yours?");  
    } // main  
} // ReadArgument
```

---

- The second line prints a thing which is called “args[0]”
- And who is this?

# Controlling (a bit) Input and Output

---

```
public class ReadArgument {  
    public static void main(String[] args) {  
        System.out.print("My favourite pets are ");  
        System.out.print(args[0]);  
        System.out.println(". And yours?");  
    } // main  
} // ReadArgument
```

---

- The second line prints a thing which is called “args[0]”
- And who is this?
- For now, to you, “args[0]” contains the first thing you typed after calling the program name when you run the code:

```
java ReadArgument sillycreature
```

# Controlling (a bit) Input and Output

---

```
public class ReadArgument {  
    public static void main(String[] args) {  
        System.out.print("My favourite pets are ");  
        System.out.print(args[0]);  
        System.out.println(". And yours?");  
    } // main  
} // ReadArgument
```

---

- The second line prints a thing which is called “args[0]”
- And who is this?
- For now, to you, “args[0]” contains the first thing you typed after calling the program name when you run the code:

```
java ReadArgument sillycreature  
will contain “sillycreature”
```

# Controlling (a bit) Input and Output

---

```
public class ReadArgument {  
    public static void main(String[] args) {  
        System.out.print("My favourite pets are ");  
        System.out.print(args[0]);  
        System.out.println(". And yours?");  
    } // main  
} // ReadArgument
```

---

- The third line prints “and yours?”
  
- So what is the result if we type  
    java ReadArgument worms  
in the console?

# Controlling (a bit) Input and Output

My favourite pets are worms. And yours?

# Controlling (a bit) Input and Output

---

```
public class ReadArgument {  
    public static void main(String[] args) {  
        System.out.print("My favourite pets are ");  
        System.out.print(args[0]);  
        System.out.println(". And yours?");  
    } // main  
} // ReadArgument
```

---

- Notice the different use of “print” and “println”

My favourite pets are worms. And yours?

# End

+++ Ende - The end - Finis - Fin - Fine +++ Ende - The end - Finis - Fin - Fine +++