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

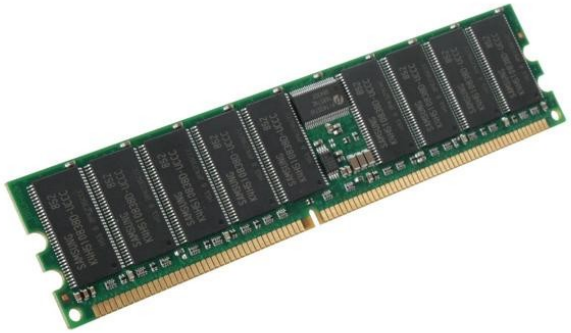
Courtesy Bundesarchiv

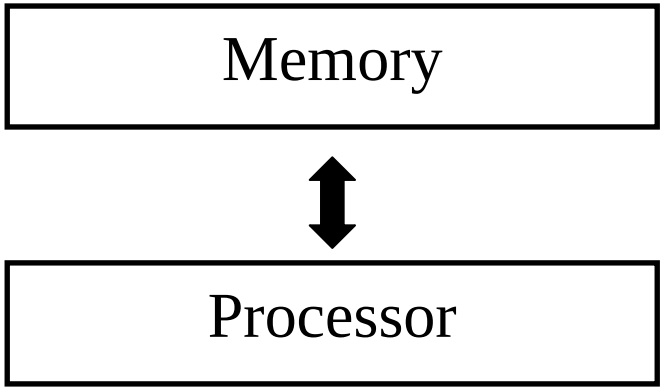
What makes a computer?

A minimal computer

Processor

A minimal computer





A minimal computer

Memory

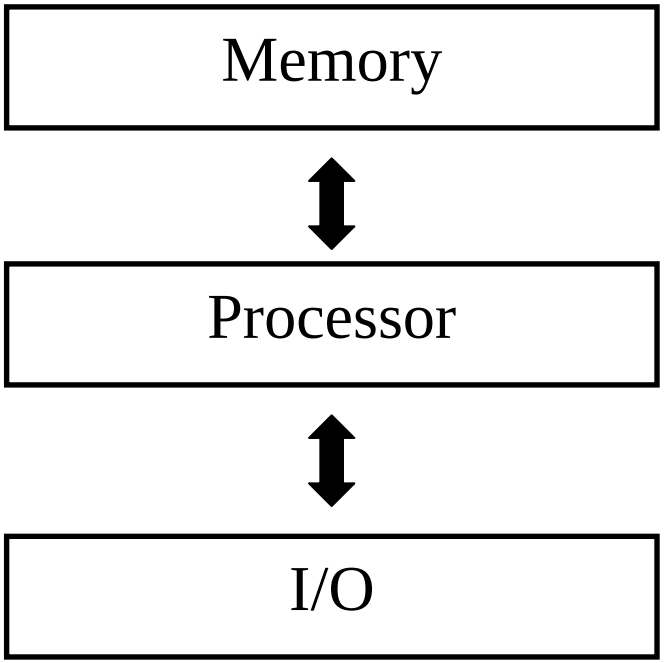
↕

Processor

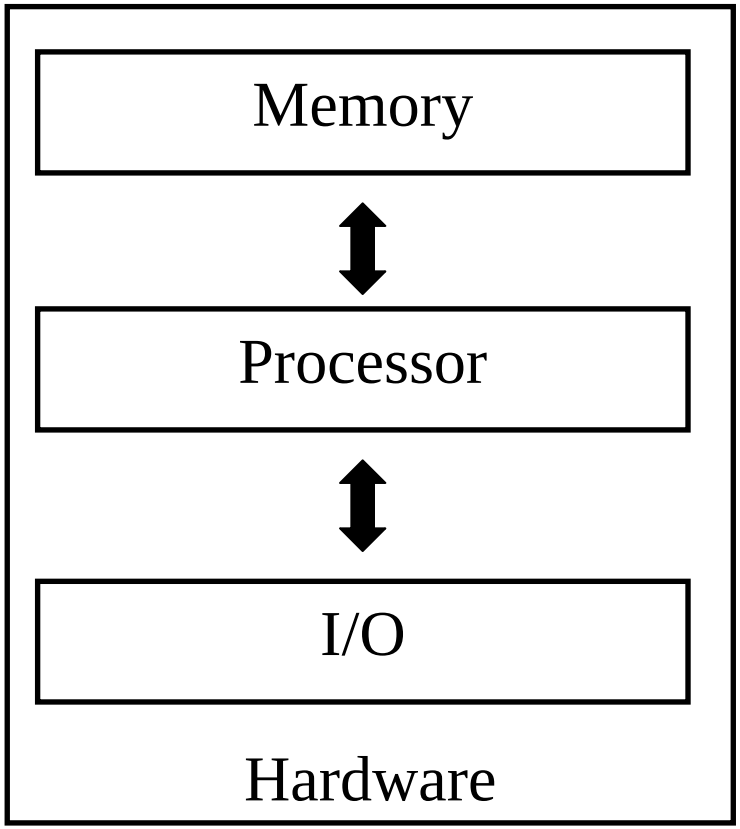
↕

I/O

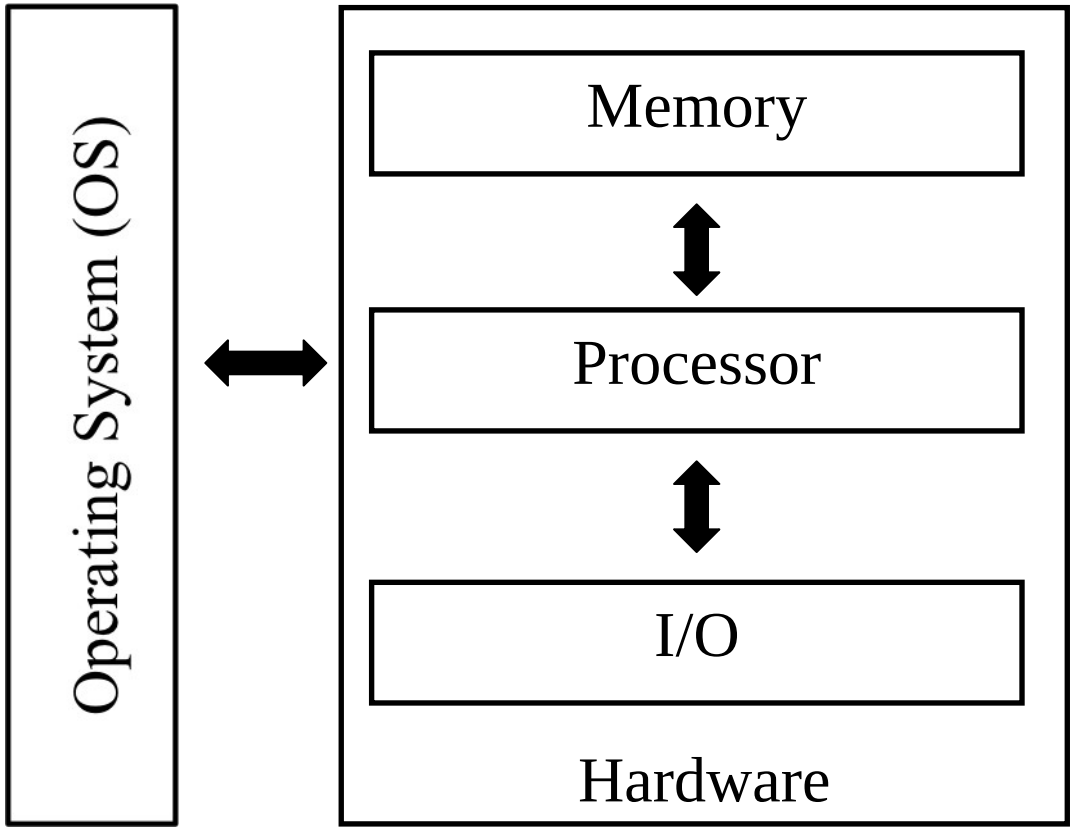
A minimal computer



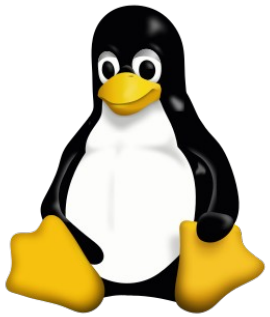
A minimal computer

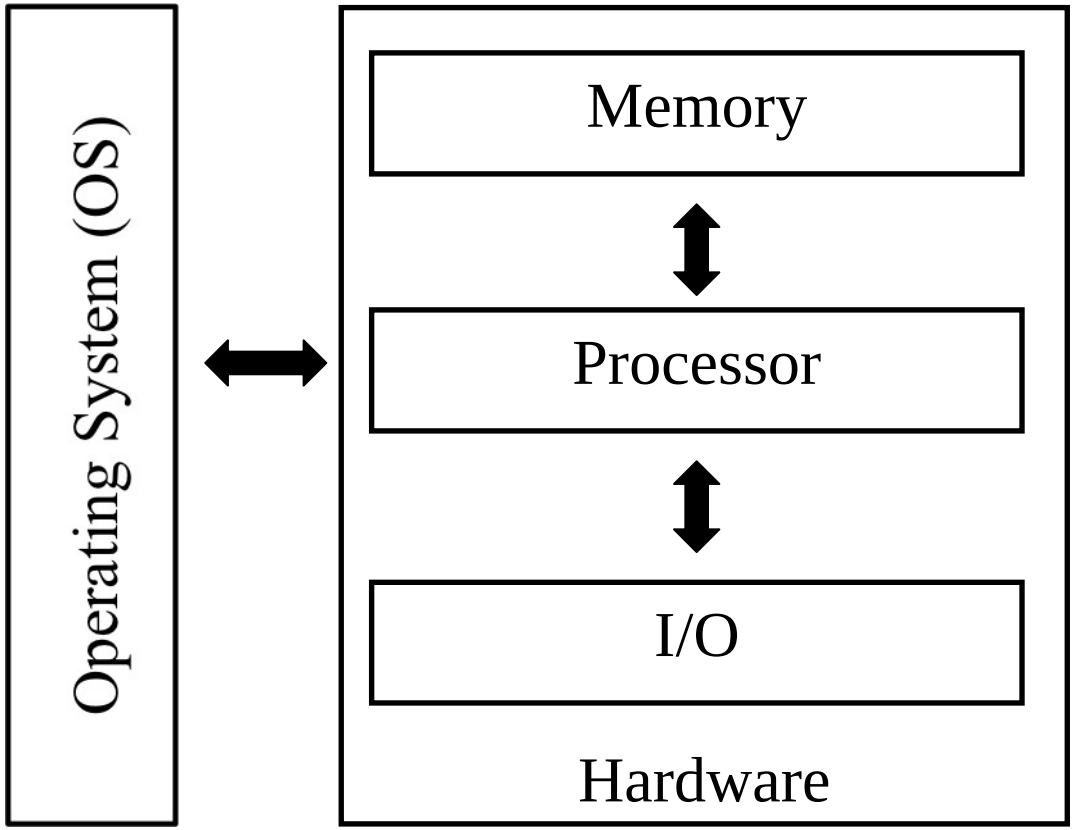


A less minimal computer

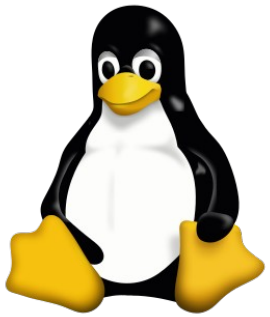


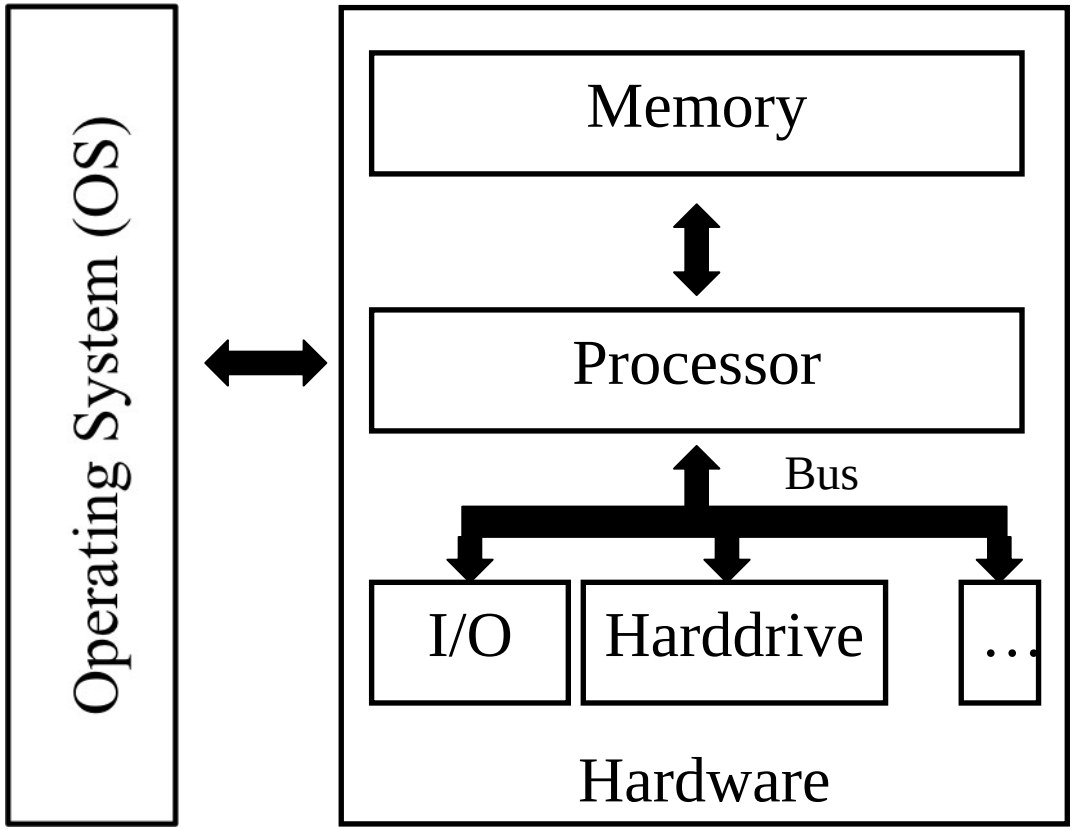
A less minimal computer



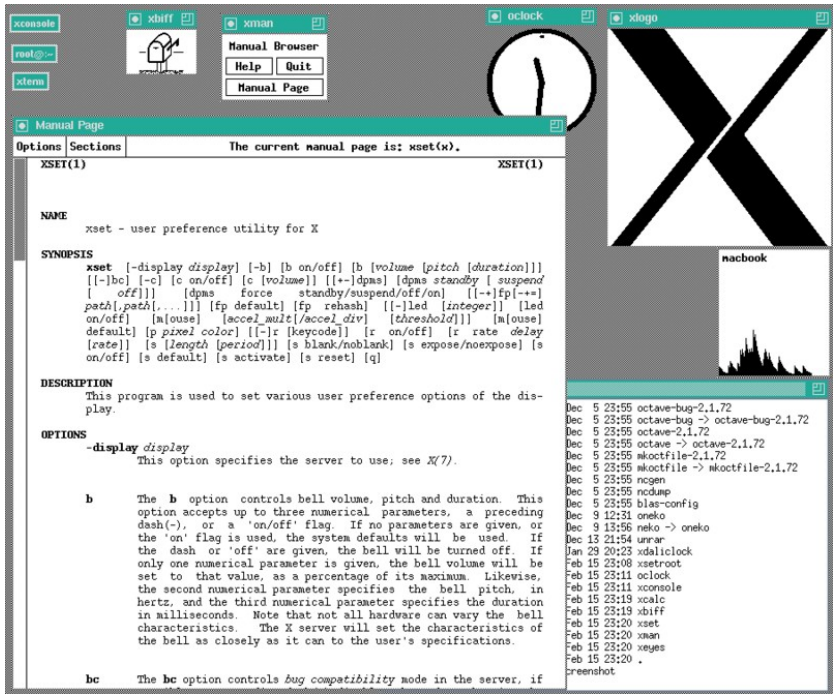


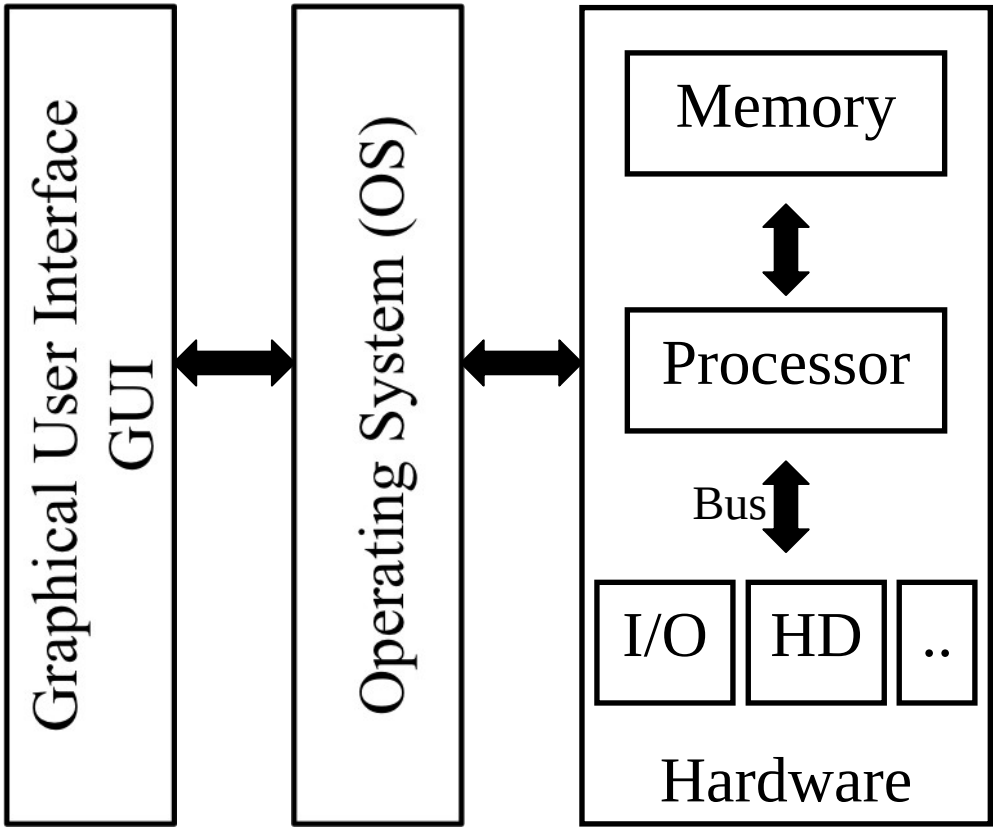
A more realistic computer



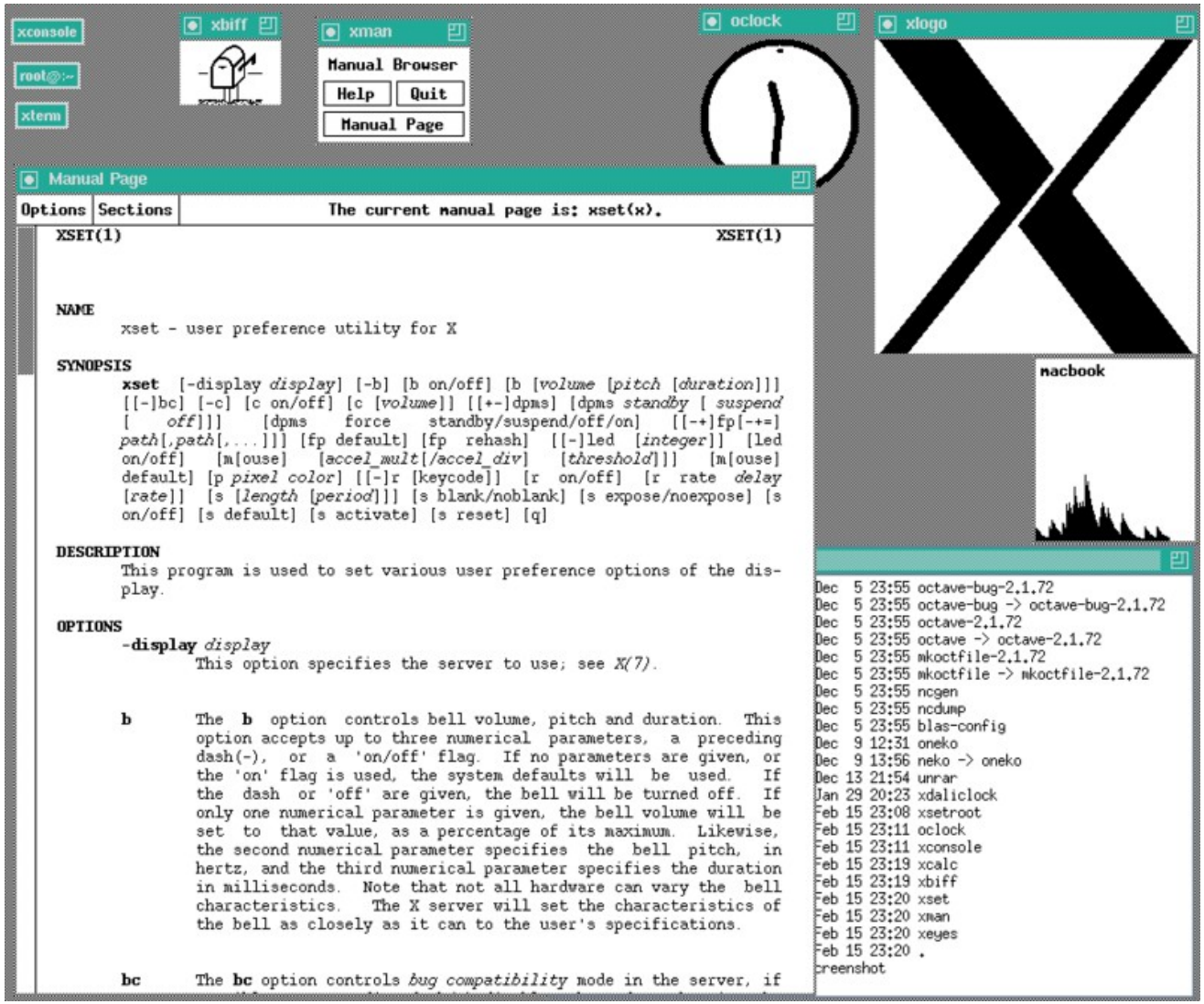


A not quite so minimal computer





A not quite so minimal computer

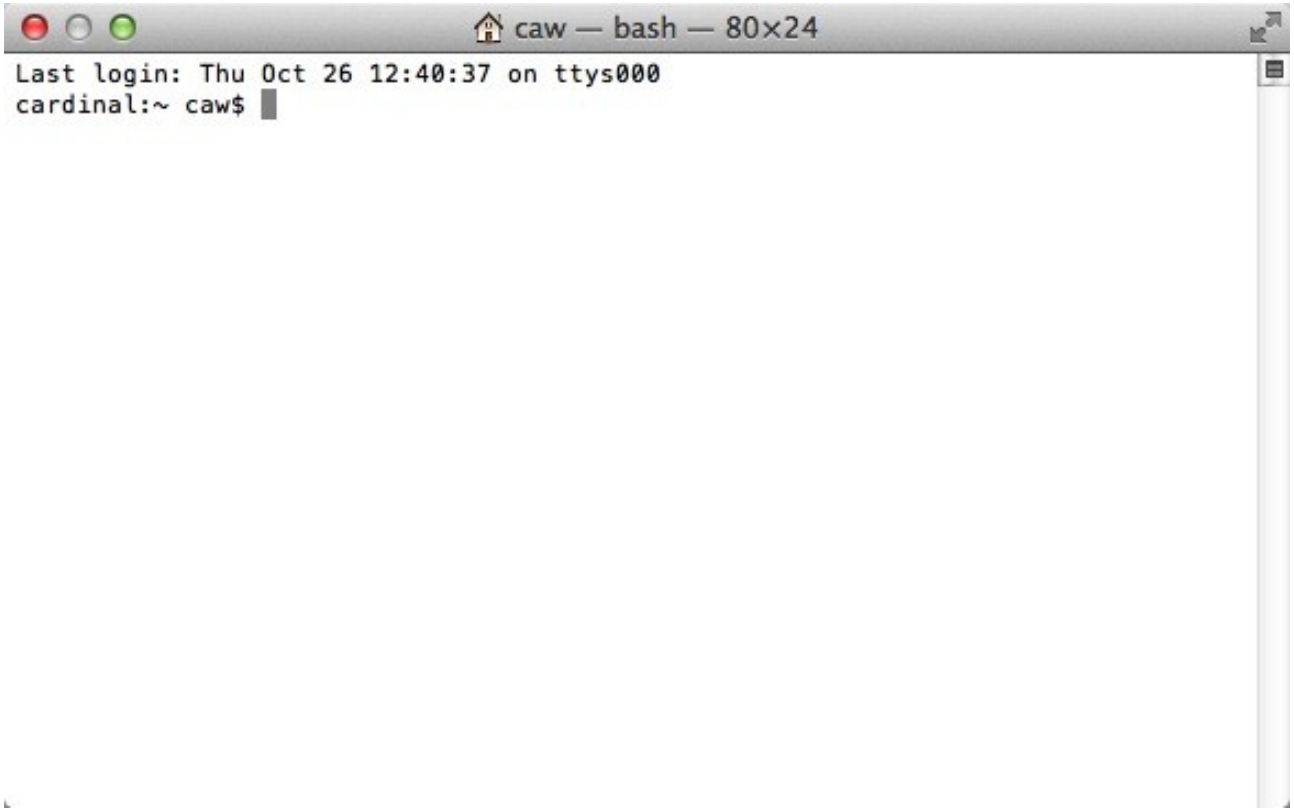


The file system

- 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



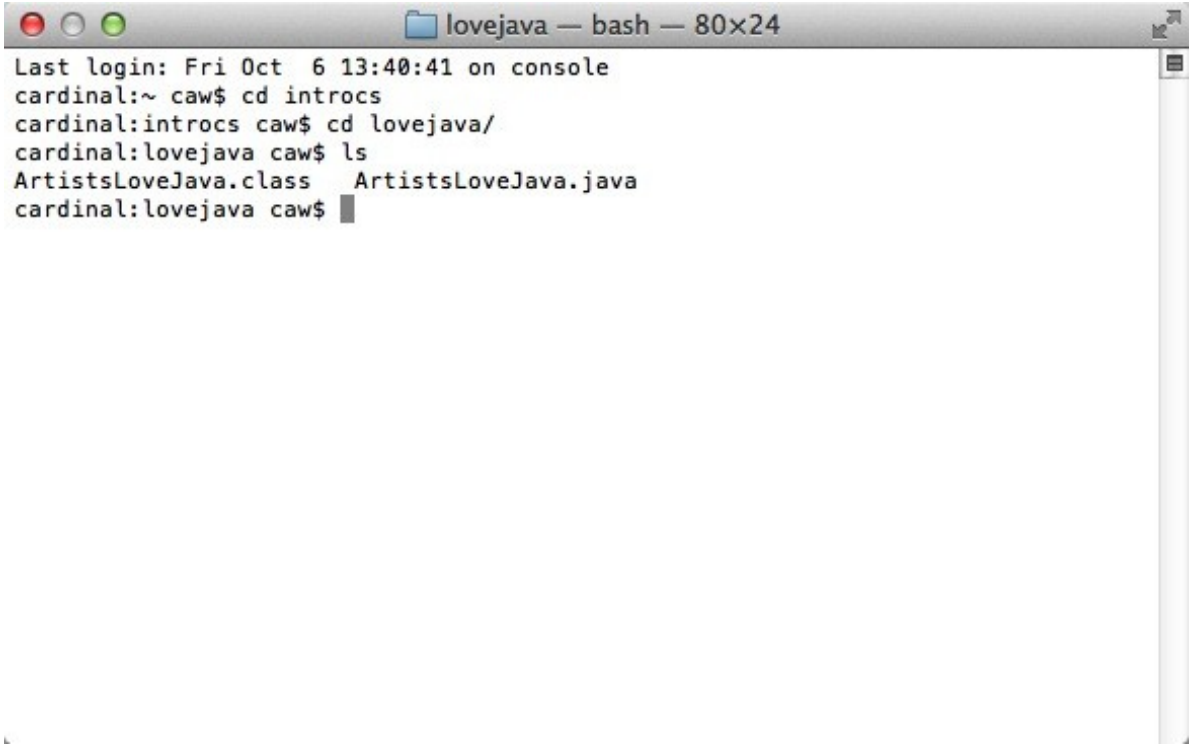
Moving in the file system

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

  `cd`

- Example:

  `cd /`



- Current directory:

  `.`

- Up directory

  `..`

  `cd ..`

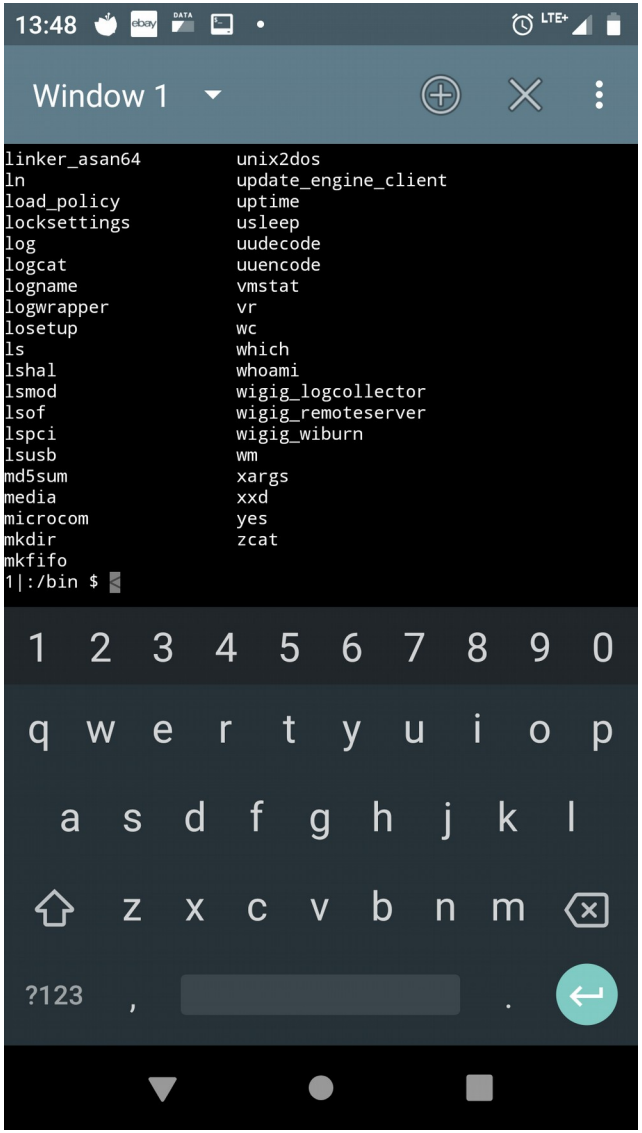
Listing files

- From any directory position:
  ```
  ls
  ```
- Example:
  ```
  ls ./introcs/
  ArtistsLoveJava.java    hello        lovejava
  checkstyle-5.5          java3d       stdlib.jar
  findbugs-2.0.3          log.txt
  ```
- Lists the subdirectory of the current directory named introcs

Listing files

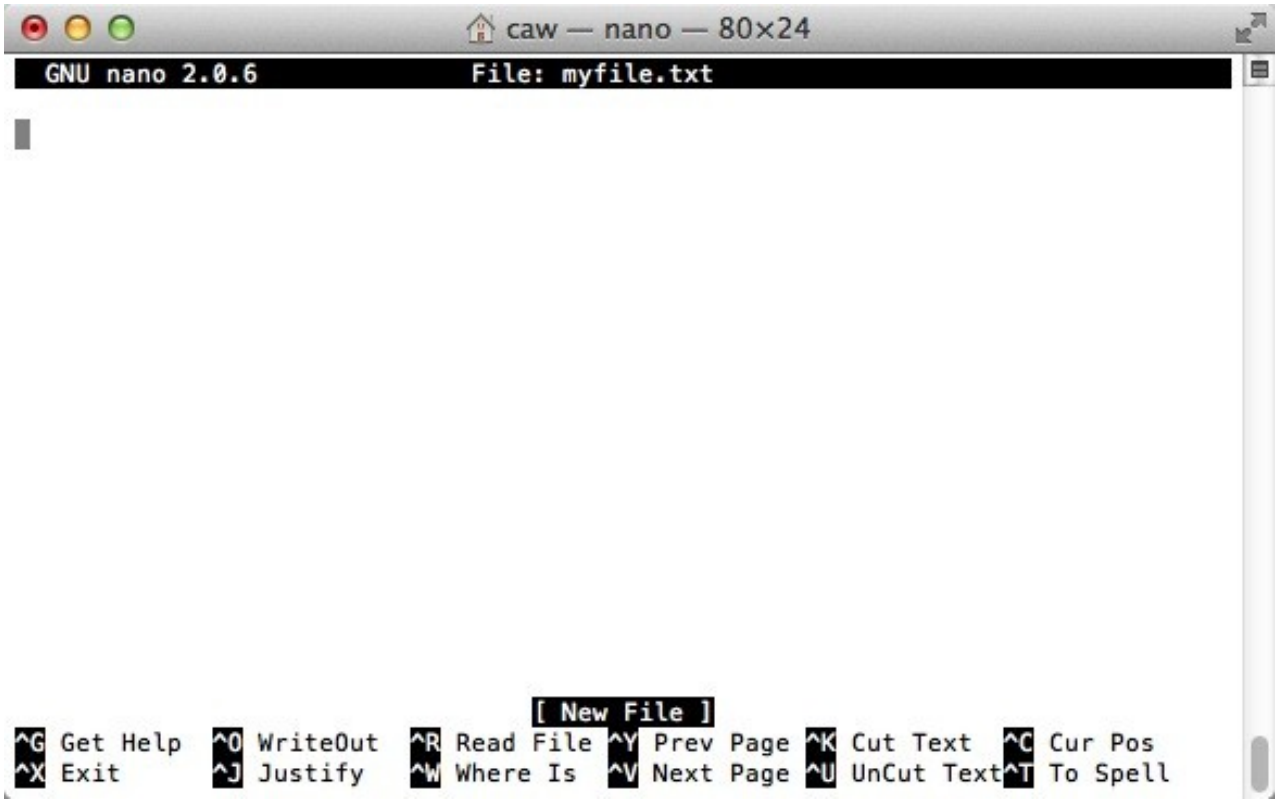
- If you think “This guy is telling me bollocks” Think again!
- Download a “Terminal Emulator” from GooglePlay
- And take a look at what is under your pretty cellphone!!!!!

- Absolutely the same!!!!



Editing files

- You will need an editor application, such as:
  `nano emacs . . . .`
- Example:
  `nano myfile.txt`



My first program

```
introcs — nano — 80×24
  GNU nano 2.0.6          File: ArtistsLoveJava.java

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

                          [ Read 5 lines ]
^G Get Help   ^O WriteOut   ^R Read File  ^Y Prev Page  ^K Cut Text   ^C Cur Pos
^X Exit       ^J Justify    ^W Where Is   ^V Next Page  ^U UnCut Text ^T To Spell
```

Compiling my first program

lovejava — bash — 80×24

```
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 now 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 — 80×24

```
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)

/Users/caw/introcs/lovejava/ArtistsLoveJava.java

New | Open | Save | Close | Cut | Copy | Paste | Undo | Redo | Find | Compile | Reset | Run | Test | Javadoc

ArtistsLoveJava.java

```
1  public class ArtistsLoveJava {
2      public static void main(String[] args) {
3          System.out.print("Artists love Java!\n");
4      } // main
5  } // class ArtistsLoveJava
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
```

Interactions | Console | Compiler Output

Compilation completed.

Compiler

JDK 6.0_65

Highlight source

Editing /Users/caw/introcs/lovejava/ArtistsLoveJava.java 5:26

My first program

```java
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

```java
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

```java
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 +++