Introduction to Programming and Computational Physics

Lesson n.1

Algorithms

Programming languages

Operating systems

Shells

The first C program

What is an algorithm?

A well-ordered and finite set of non-ambiguous and computable operations that leads to a result and terminates in a finite time when applied to a set of initial conditions

A well-written algorithm

The recipe for cooking 100 g of pasta:

1) Put 1 liter of water in a pot
2) Put the pot on to cook
3) Switch on the kitchen stove
4) Repeat step n.5 until the water starts to boil
5) Wait one minute
6) Add 10 g of salt
7) Read the cooking time on the pasta envelop
8) Put the pasta in the boiling water
9) Wait the time given at step n.7
10) Strain your pasta
11) End

A badly-written algorithm

$$$

An algorithm to earn money at the Stocks Exchange:

1) If the stocks lowered in price in such a way that they can only increase their value… **BUY**

2) If the stocks increased in price in such a way that they can only reduce their value… **SELL**

What is wrong with it?

The greatest common divisor (Euclid III century b.C)

1) Take two numbers
2) Evaluate the remainder of the integer division (the greater number over the smaller one)
3) If the remainder is zero ***go to*** 6)
4) Replace the greater number with the remainder of the division
5) ***Go to*** 2)
6) The smaller number is the greatest common divisor
7) End

This approach with “jumps” between instructions is not used anymore in the so-called *structured programming*

Structured programming

The idea is to execute all the instructions in a sequential way from the beginning to the end of program with two only *exceptions* allowed

- conditional operations
  if a condition is verified *do something* (optional: otherwise *do something else*)

- iterative operations
  as long as a condition is verified *do something*

In 1966 Corrado Böhm and Giuseppe Jacopini demonstrated that using condition and iteration all the programs can be written without using *goto*

Why algorithms are so important?

Our aim is to build one or more computation instruments able to execute “primitive operations” (…and computable)

The solution of a problem expressed by an algorithm made up of a sequence of primitive operations can be *automated*

A ***program*** **is the realization of one or more algorithms with a sequence of primitive operations understood by the *executor***

Programming languages

An algorithm written in a *natural* language (English, German, Italian) can’t be executed from a computer: we need a *formal* language. It must be a language provided with a set of rules in order to avoid any possible ambiguity.

A program is actually an algorithm written in a formal language.

The C language

1969:

Ken Thompson (Bell Telephone) wrote the B language: a first attempt to define a high-level language for operating system implementation.

1972:

Dennis Ritchie wrote an evolution of the B language: the C language. The UNIX operating system was almost entirely written in C.

1973-1980:

The C language spreads to many other platforms. The first *libraries* are born and the first reference book is written in 1978: Kernighan & Ritchie,"C Programming Language“.

1983 - 1999:

The American National Standards Institute defines the standard ANSI C: a collection of rules to be followed by any C compiler.

Operating system

An **operating system** (**OS**) is a set of computer programs that manage the hardware and software resources of a computer. Its basics tasks are:

- Processing management
- Memory management
- Recognizing Input and sending Output
- Controlling peripheral drivers
- Networking

They provide a *software platform* on top of which other programs, called *application programs* can run

The most popular: Microsoft Windows (MS-DOS)

Unix/Linux

Mac OS X

Shell

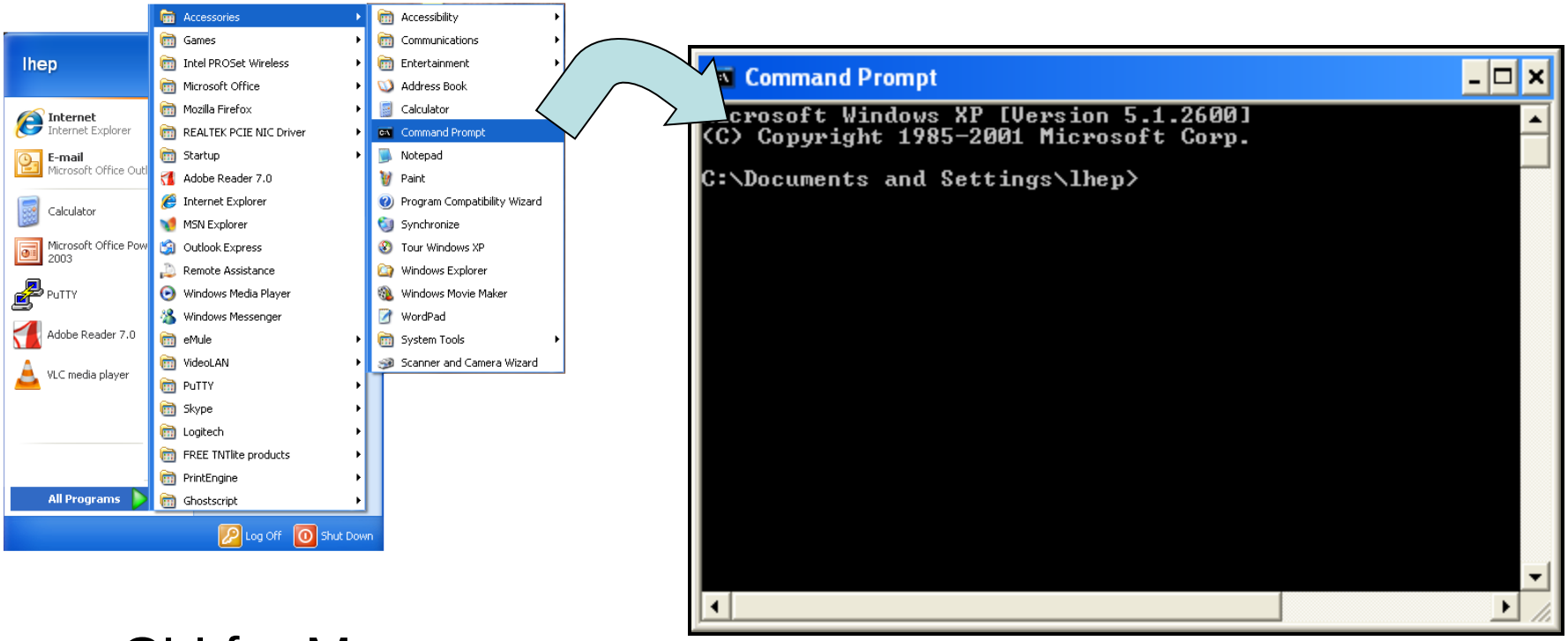
An *operating system shell* is a software that provides an interface between users and the OS.

Basic features:

- to invoke (or "launch“) another program
- viewing the contents of directories
- copying/moving files

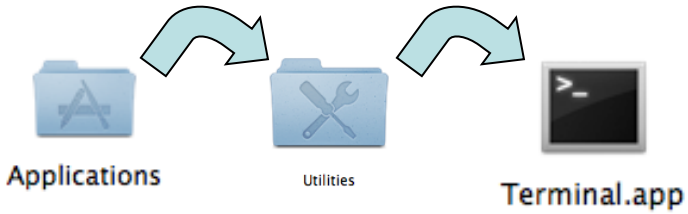
It can work as *command line interface* (CLI) or *graphical user interface* (GUI)

CLI for Windows



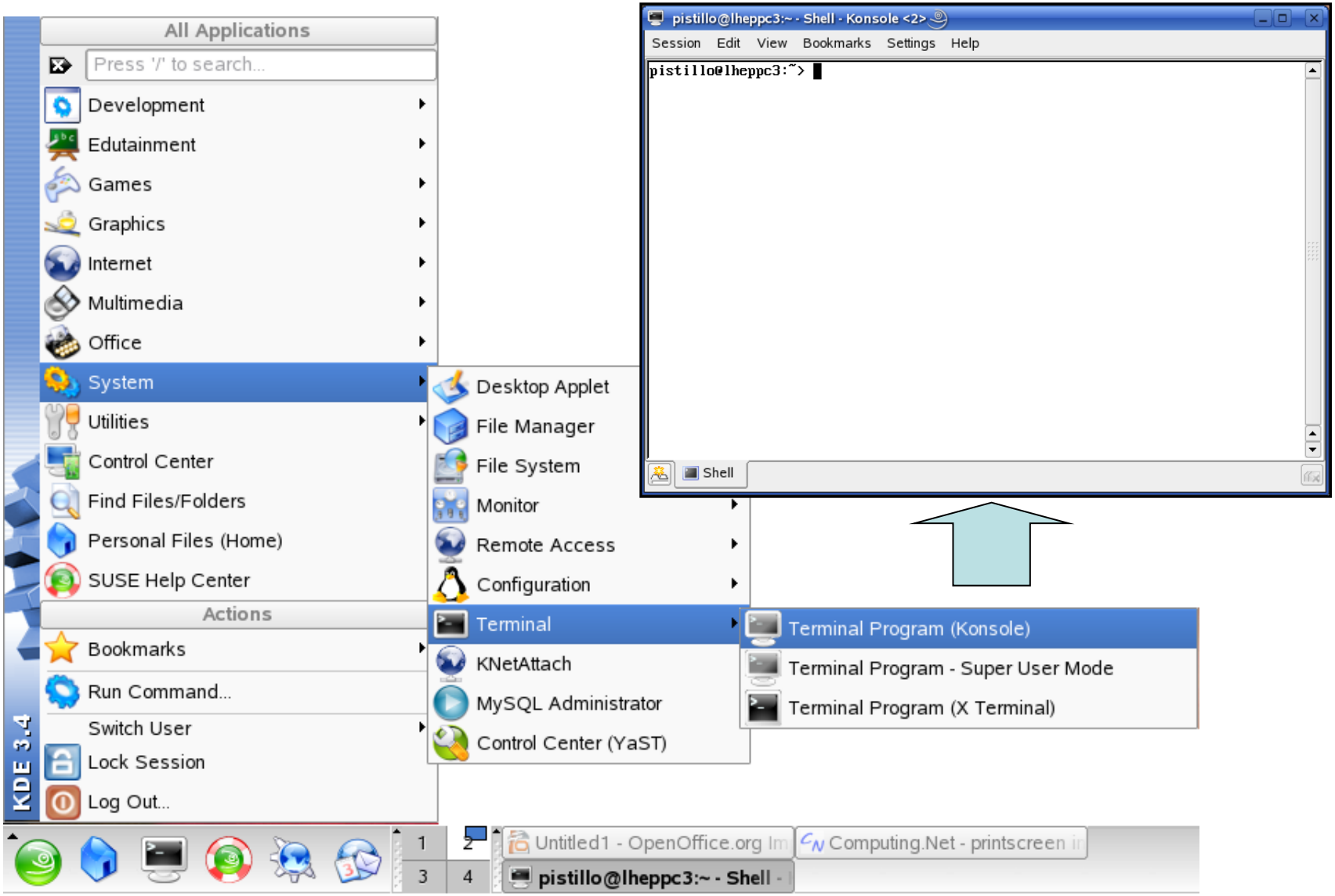
CLI for Mac

Use the Spotlight Search in the upper right corner and search for "Terminal". OR ….

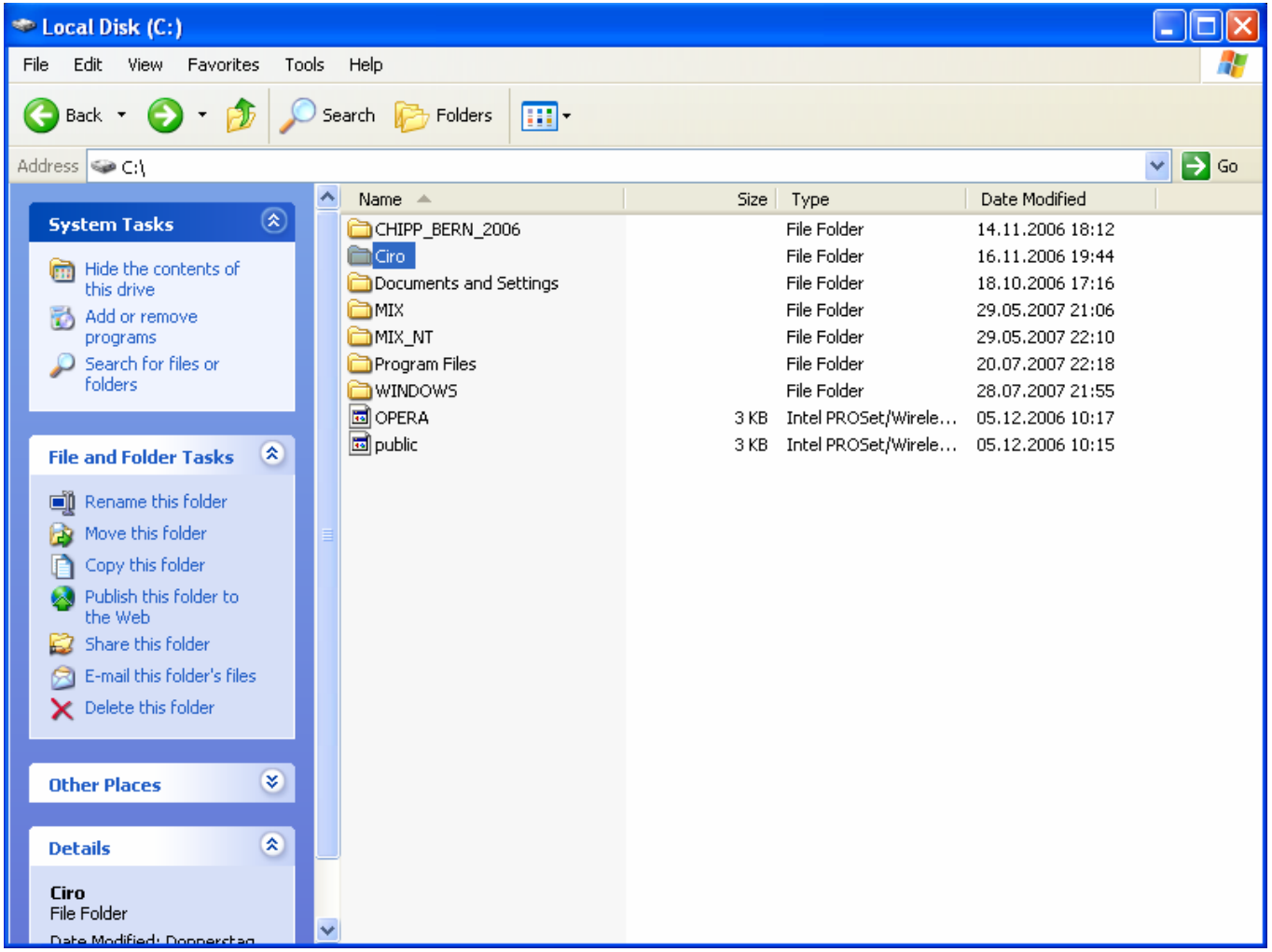


Macintosh HD -> Applications -> Utilities -> Terminal
(German: Macintosh HD -> Programme -> Dienstprogramme -> Terminal)

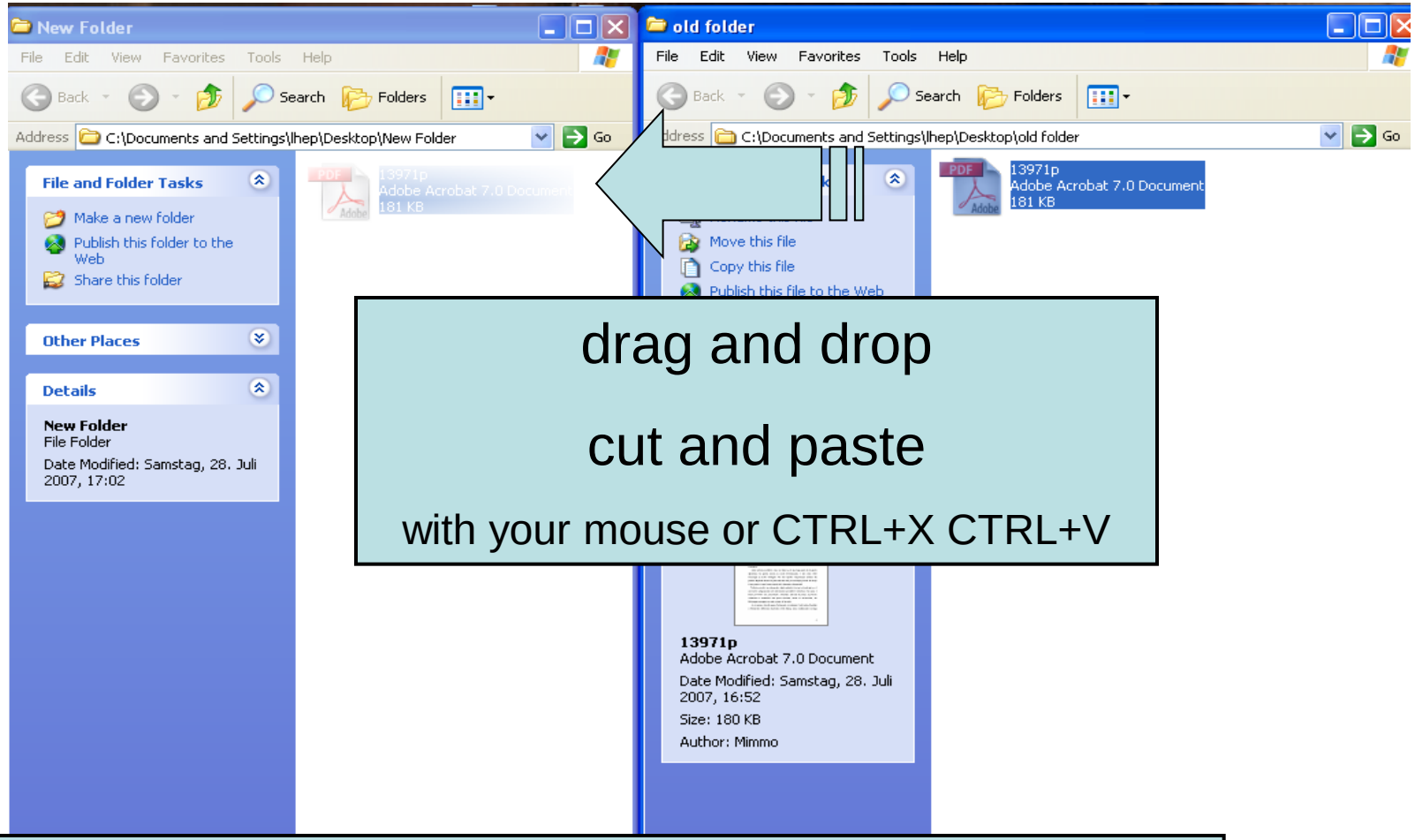
CLI for Linux (KDE)



Windows Explorer: a GUI for Windows



Using a GUI



drag and drop

cut and paste

with your mouse or CTRL+X CTRL+V

Faster and easier(?) but… don't loose the control of what you're doing…

Text files

A text file is a sequence of characters from a recognized character set (ASCII, Unicode).

-Plain text (only text, newline codes, “end of file” markers)

-Structured text (many additional informations: markers for **bold** or *italic* start/end, colored text, paragraphs start/end, numbered list…)

```
 !"#$%&'()*+,-./
0123456789:;<=>?
@ABCDEFGHIJKLMNO
PQRSTUVWXYZ[\]^_
`abcdefghijklmno
pqrstuvwxyz{|}~
```

The American Standard Code for Information Interchange, based on the English alphabet, is a character encoding (95 printable characters numbered from 32 to 126).

It specifies a correspondence between digital 7-bit patterns and symbols of a written language

text editor

It is a program for text file editing. They are usually provided with the OS.

Windows: notepad, wordpad, word, notepad++

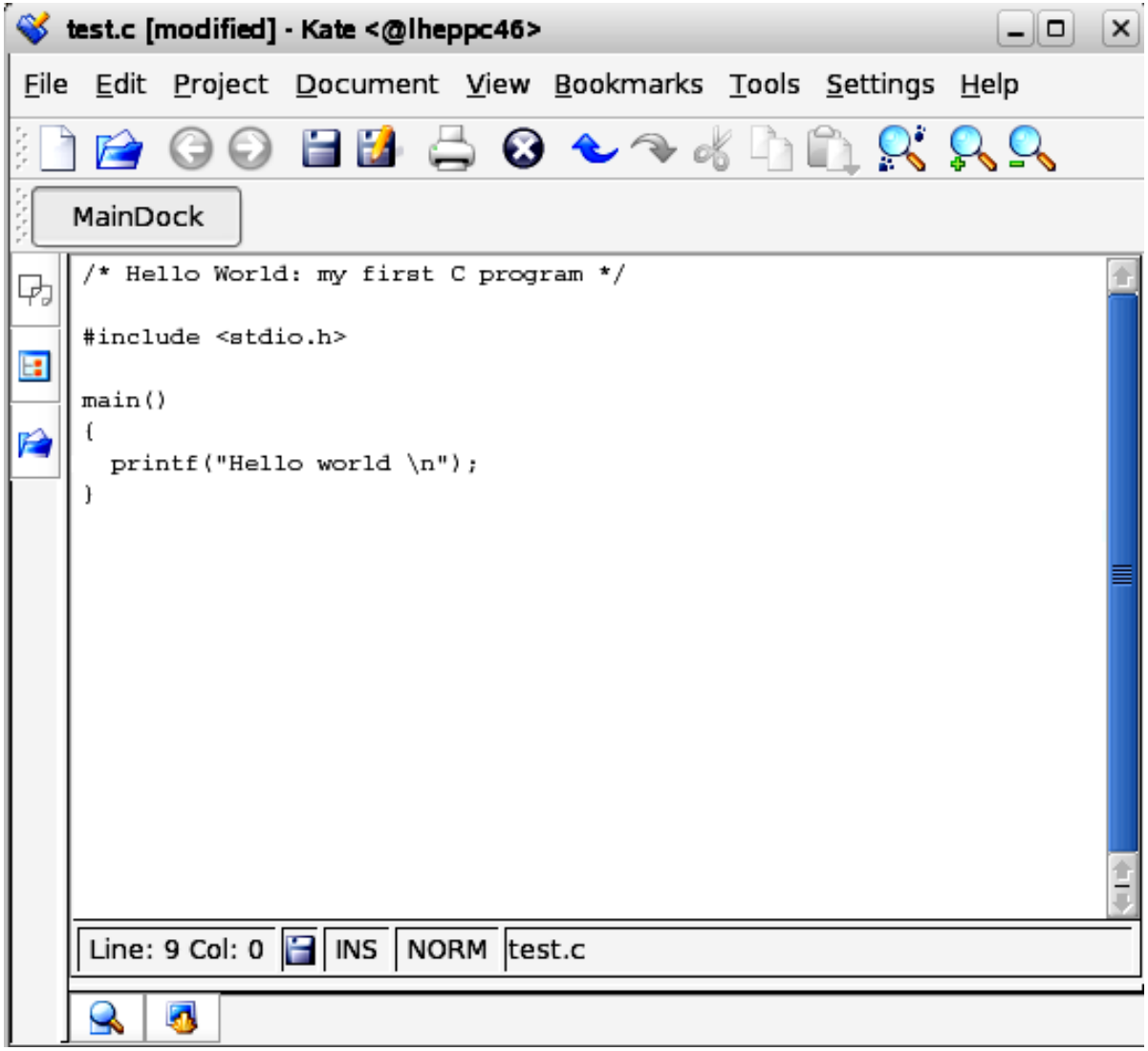
Linux: vi, emacs, kate

Mac: Xcode, TextWrangler

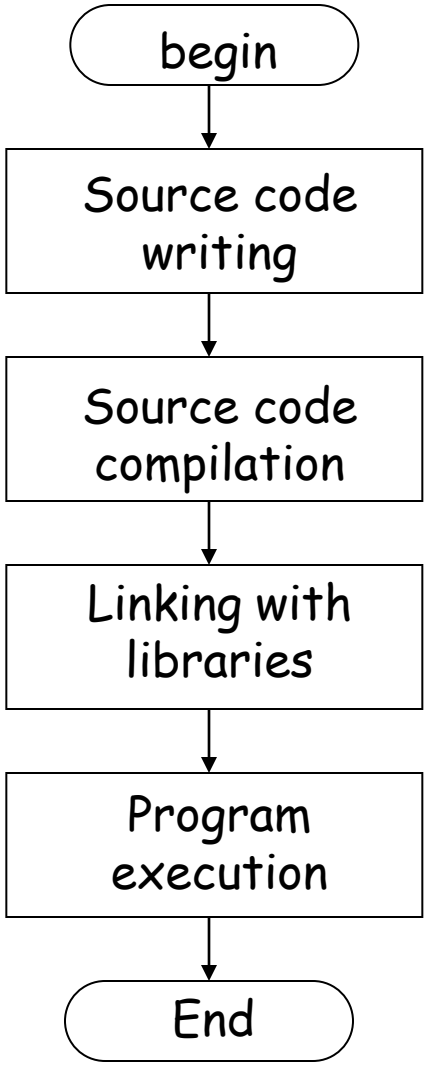
Some of them are designed for writing the program language source code. Typical features:

- search and replace
- copy, cut and paste
- text formatting (indentation)
- undo and redo
- syntax checks
- …

Kate: an editor for Linux



Development of a C program



	(gcc compiler):	**Effects:**
Source code writing	edit…write… prog.c	Source file written
Source code compilation	> gcc -c prog.c	Object file prog.o written
Linking with libraries	> gcc -o prog.exe prog.o	Executable file prog.exe written
Program execution	> prog.exe (or ./prog.exe)	The OS executes the instructions given by the program

You can also directly type: > gcc –o prog.exe prog.c **(compilation+linking)**

The first C program

```
/* This is my first C program,
   just to say hallo to the world */

// I include the I/O library

#include <stdio.h>

int main() //this is the main function
{
  printf("Hallo, World");
  printf("\n");

  return 0;
}
```

Comments: the compiler ignore what is between /* and */ or what follows // (till the end of the row)

All the instructions end with a semicolon

start/end of a block (here, the main function)