



AISF - COMITATO LOCALE DI PERUGIA

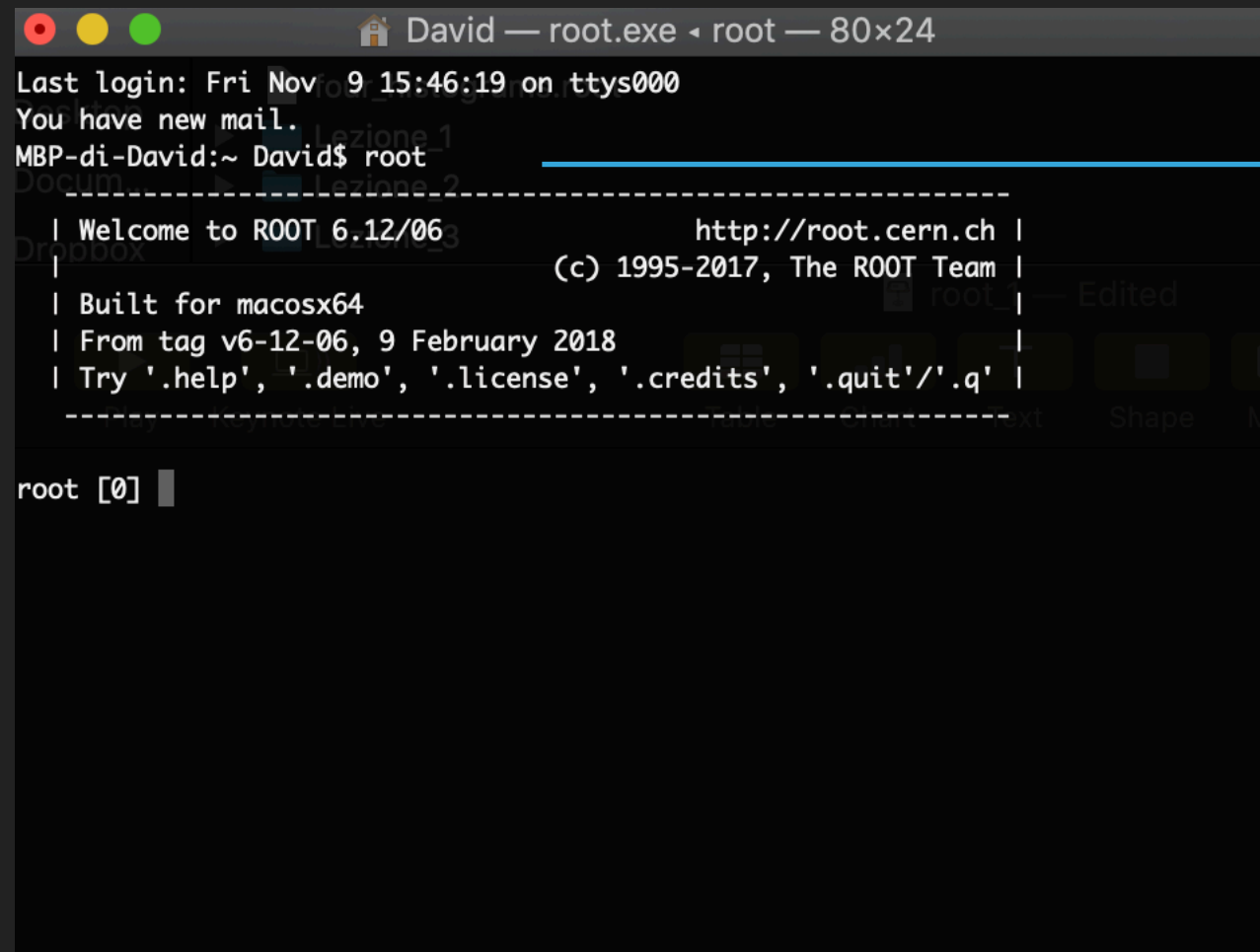
CORSO ROOT

INTRO COMANDI BASE ROOT

► Informazioni Generali

- Root è un pacchetto software per analisi dati fornito dal Cern, contenente una serie di funzioni raggruppate in Classi (**TGraph**, **TGraphErrors**, **TMultigraph**,.....)
- Ogni classe ha una serie di funzioni corrispondenti chiamate **metodi** della classe. Ogni metodo riceverà in input una serie di parametri.
- se volessimo ad esempio creare un grafico con errori sperimentali si dovrà creare un **oggetto** appartenente alla classe corrispondente.
- Nelle prossime slides vediamo come creare gli oggetti grafici fondamentali per eseguire analisi dati di livello base. Come già anticipato vedremo come creare grafici per punti sperimentali, eseguire fit sul set di dati graficati, creare istogrammi ed eseguire fit gaussiani. Ognuno di questi oggetti apparterrà dunque ad una classe corrispondente e i vari metodi che applicheremo agli oggetti dovranno essere definiti da parametri di input specifici.

- ▶ una volta installato correttamente si può accedere al software root dal terminale digitando il comando: **“root”**.



```
David — root.exe ◀ root — 80x24
Last login: Fri Nov 9 15:46:19 on ttys000
You have new mail.
MBP-di-David:~ David$ root
-----
| Welcome to ROOT 6.12/06                               http://root.cern.ch |
| (c) 1995-2017, The ROOT Team                           |
| Built for macosx64                                     |
| From tag v6-12-06, 9 February 2018                     |
| Try '.help', '.demo', '.license', '.credits', '.quit'/.q' |
|-----|
root [0]
```

comando di
apertura
software

- ▶ vediamo come creare un oggetto grafico per visualizzare la funzione **“sin(x)/x”**. La classe specifica per tale scopo sarà la classe TF1,

▶ analizziamo i comandi

```
David — root.exe ◀ root — 80x24
Last login: Fri Nov 9 15:46:19 on ttys000
You have new mail.
MBP-di-David:~ David$ root
-----
| Welcome to ROOT 6.12/06             http://root.cern.ch |
|                                     (c) 1995-2017, The ROOT Team |
| Built for macosx64                 |
| From tag v6-12-06, 9 February 2018 |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.'q' |
|                                     |
-----

root [0] TF1 f1("f1", "sin(x)/x",0,10)
(TF1 &) Name: f1 Title: sin(x)/x
root [1] f1.Draw()
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
root [2] 
```

TF1:nome classe dell'oggetto

f1:nome oggetto

parametri (.....)

- ▶ Nel caso semplice di grafico di funzione analitica nota i parametri sono:
- ▶ espressione analitica: "sin(x)/x"
- ▶ range del dominio (0,10)
- ▶ f1 sarà il nome dell'oggetto appartenente alla classe TF1

- ▶ per visualizzare l'oggetto f1 appena creato usiamo il metodo (comune a quasi tutte le classi) **Draw()**.

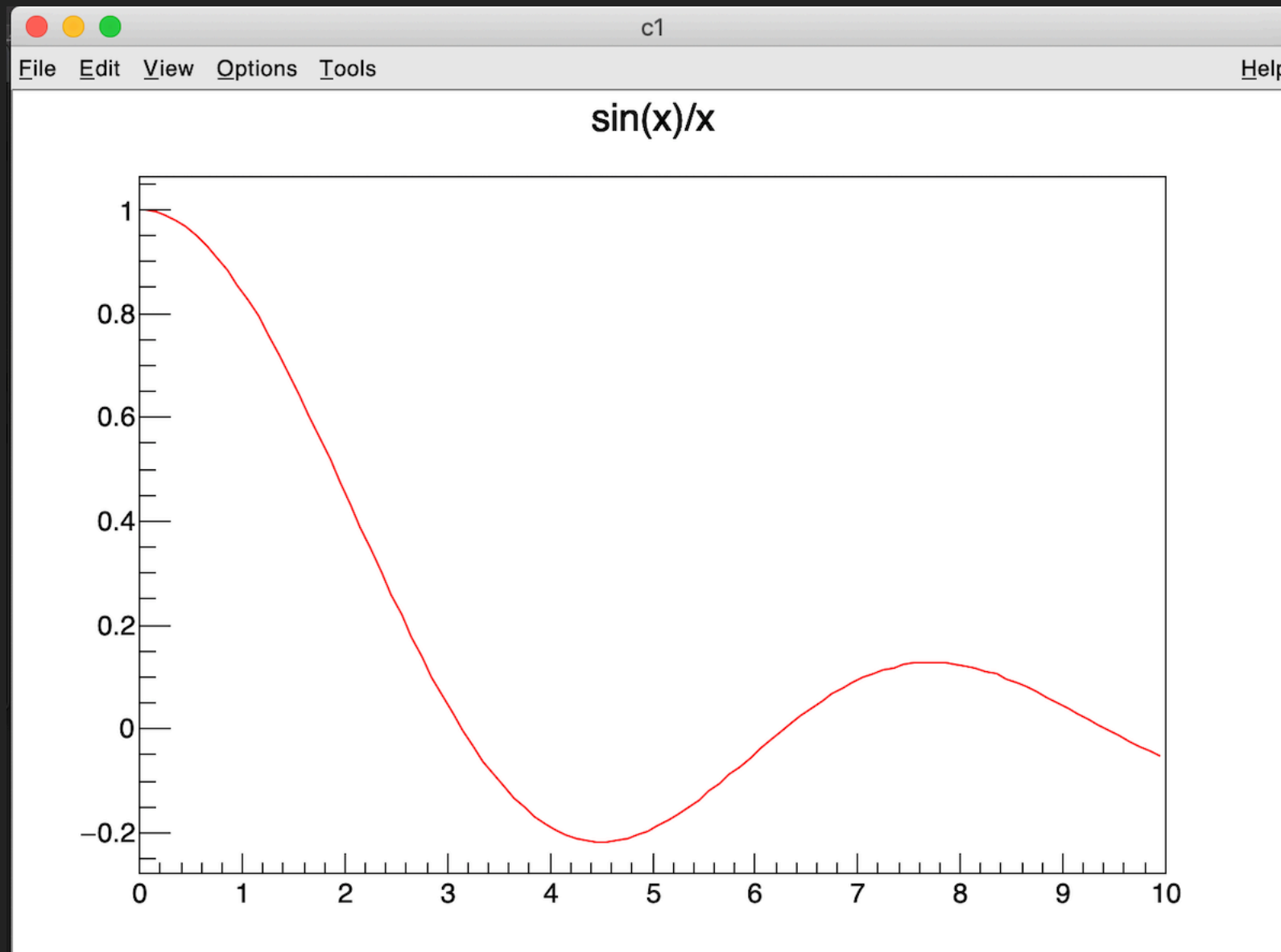
```
David — root.exe ◀ root — 80x24
Last login: Fri Nov  9 15:46:19 on ttys000
You have new mail.
[MBP-di-David:~ David$ root]
-----
| Welcome to ROOT 6.12/06             http://root.cern.ch |
|                                     (c) 1995-2017, The ROOT Team |
| Built for macosx64                  |
| From tag v6-12-06, 9 February 2018 |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.'q' |
|                                     |
-----

[root [0] TF1 f1("f1", "sin(x)/x",0,10)
(TF1 &) Name: f1 Title: sin(x)/x
[root [1] f1.Draw()
Info in <TCanvas::MakeDefCanvas>:  created default TCanvas with name c1
root [2] ]
```

premando invio, creiamo correttamente il grafico

visualizzo il grafico

- ▶ con il comando **Draw()** questo è ciò che appare, ossia la canvas contenente il grafico nel range scelto.



- ▶ con lo stesso ragionamento possiamo creare anche grafici più interessanti come grafici con punti sperimentali(e relativi errori) .
- ▶ useremo un classe diversa: TGraphErrors.
- ▶ prima dobbiamo creare gli array contenenti i punti sperimentali e i relativi errori.
- ▶ in particolare 4 array: (dati x , dati y , errori su x , errori su y)

creazione singola
dei vari
array(double) di
dimensione 3

```
David — root.exe ◀ root — 80x24
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$
MBP-di-David:~ David$ root
-----
| Welcome to ROOT 6.12/06                      http://root.cern.ch |
|                                              (c) 1995-2017, The ROOT Team |
| Built for macosx64                          |
| From tag v6-12-06, 9 February 2018          |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.'q' |
-----

root [0] double x[3] = {0.2 , 0.4,0.6}
(double [3]) { 0.20000000, 0.40000000, 0.60000000 }
root [1] double y[3] = {0.1 , 0.5,0.8}
(double [3]) { 0.10000000, 0.50000000, 0.80000000 }
root [2] double erry[3] = {0.01 , 0.01,0.01}
(double [3]) { 0.01000000, 0.01000000, 0.01000000 }
root [3] double errx[3] = {0.01 , 0.01,0.01}
(double [3]) { 0.01000000, 0.01000000, 0.01000000 }
root [4]
```


- ▶ chiamiamo la classe **TGraphErrors**, creando l'oggetto **g**
- ▶ vediamo i nuovi parametri: **(3,x,y,errx,erry)**
- ▶ **3** = lunghezza degli array
- ▶ **(x,y,errx,erry)** rispettivamente gli array contenuti dati e errori

```
David — root.exe — root — 80x24
[MBP-di-David:~ David$
[MBP-di-David:~ David$
[MBP-di-David:~ David$ root

-----
| Welcome to ROOT 6.12/06                      http://root.cern.ch |
|                                           (c) 1995-2017, The ROOT Team |
| Built for macosx64                        |
| From tag v6-12-06, 9 February 2018        |
| Try '.help', '.demo', '.license', '.credits', '.quit'/'.'q' |
-----

[root [0] double x[3] = {0.2 , 0.4,0.6}
(double [3]) { 0.20000000, 0.40000000, 0.60000000 }
[root [1] double y[3] = {0.1 , 0.5,0.8}
(double [3]) { 0.10000000, 0.50000000, 0.80000000 }
[root [2] double erry[3] = {0.01 , 0.01,0.01}
(double [3]) { 0.01000000, 0.01000000, 0.01000000 }
[root [3] double errx[3] = {0.01 , 0.01,0.01}
(double [3]) { 0.01000000, 0.01000000, 0.01000000 }
[root [4] TGraphErrors g(3,x,y,errx,erry)
(TGraphErrors &) Name: Graph Title: Graph
[root [5] g.Draw("AP")
Info in <TCanvas::MakeDefCanvas>: created default TCanvas with name c1
[root [6] ]
```

con il metodo **g.Draw()** visualizzo il grafico

► **Compattare i Comandi: Macros**

**TUTTI I COMANDI VISTI PER GENERARE IL GRAFICO:
CREATION ARRAY E CHIAMATA DELL'OGGETTO TGRAPHERRORS,
POSSONO ESSERE IN REALTÀ ACCORPATI IN UN SOLO COMANDO
CREANDO UNA MACROS (OSSIA UNA COLLEZIONE DI COMANDI ROOT
CHE VENGONO ESEGUITI IN SINGOLA SESSIONE)**

in allegato trovate un esempio di macros reale(riferita ad una vera esperienza di Lab1) sulla creazione di un grafico con dati sperimentali e relativi errori.

Creare una macros risulta molto comodo per velocizzare l'esecuzione dei comandi.

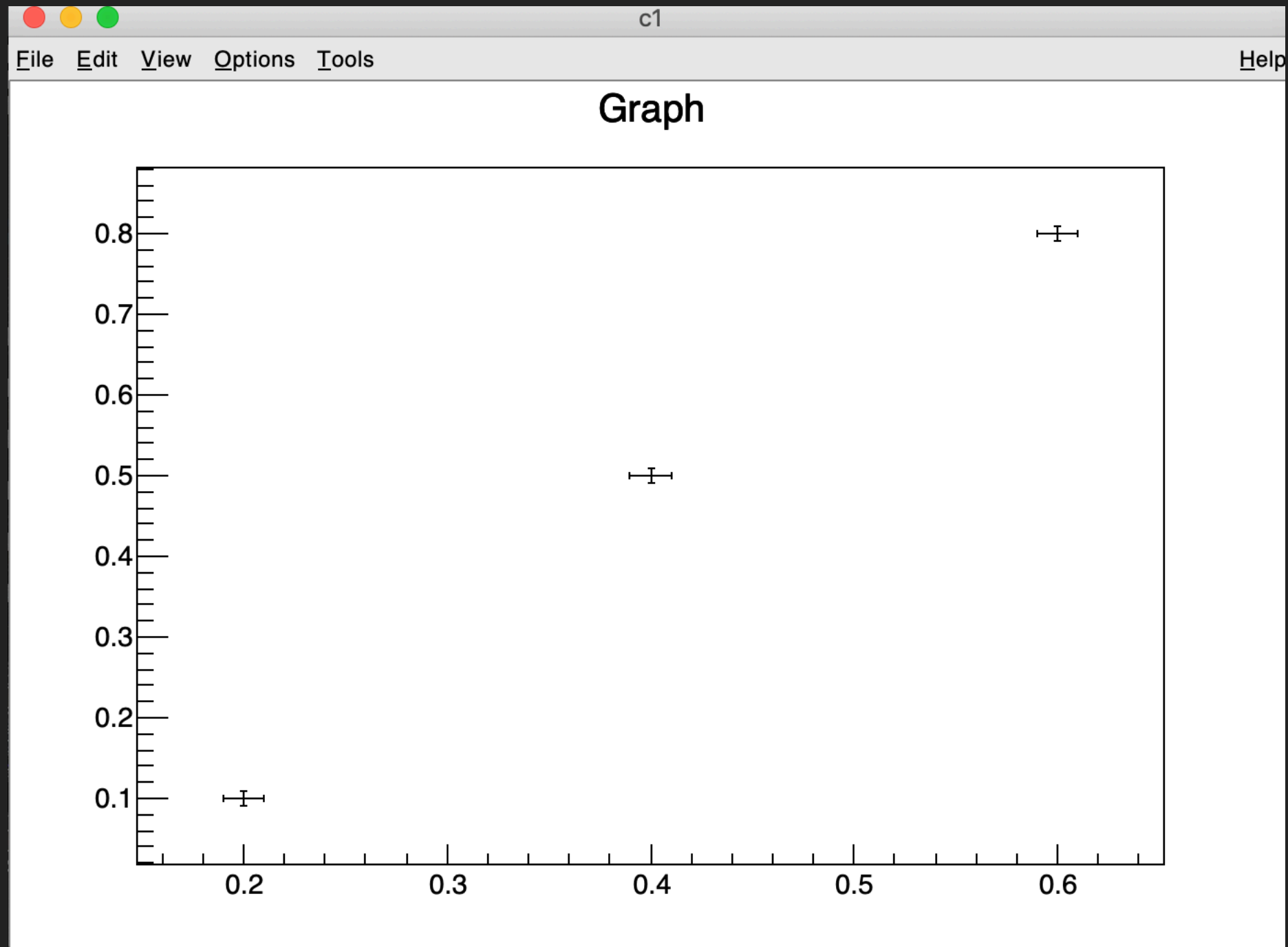
Salvando la macros in un path specifico, dipende dal tipo di installazione di Root eseguita sulla vostra macchina,la macros può essere eseguita attraverso il comando:

.x Nomemacros

es: Graph.cpp (la macros allegata)

.x Graph.cpp

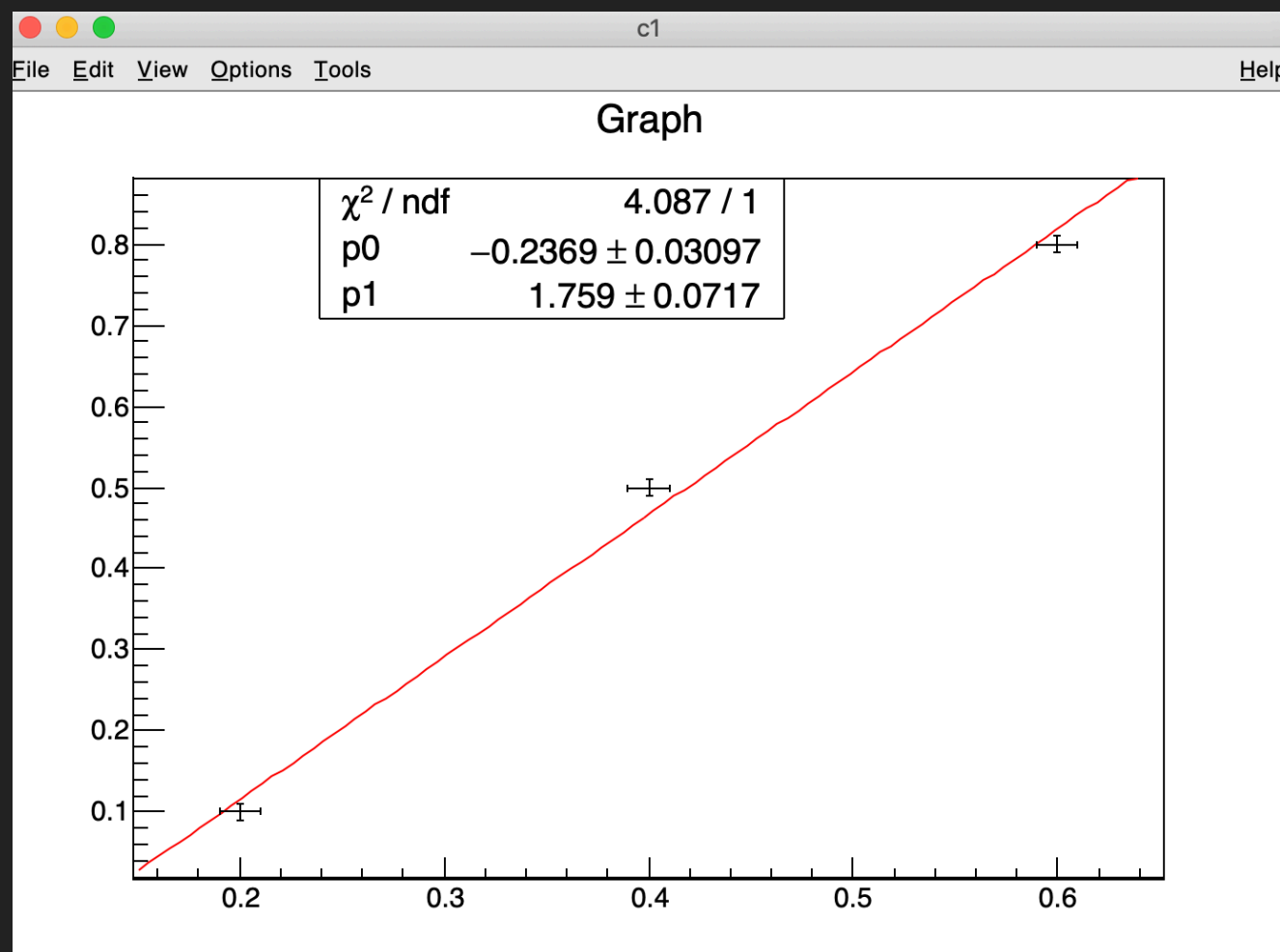
► questo è ciò che viene visualizzato



► i 3 punti sperimentali con i relativi errori su asse x e asse y

► Fit

- per effettuare un fit lineare esistono metodi specifici della classe TGraphErrors tuttavia una volta disegnato il grafico con Draw() si possono ottenere i dati di qualsiasi fit seguendo queste indicazioni:
- dalla schermata di root (Canvas) che si apre: **Tools -> fit panel->** impostare **pol1**(polinomiale grado 1 ossia retta) dal pannello **Fitfunction**
- per visualizzare i risultati del Fit Lineare sul grafico : **option -> Fit parameter**



Nota: Root indirettamente chiama dei metodi della classe TGraphErrors. Tuttavia la soluzione "dinamica" senza modificare la macros è la più comoda per molte applicazioni . Se lo scopo fosse quello di estrarre il valore del coefficiente p1 del fit per manipolarlo all'interno della macros la soluzione ovvia è quella di applicare il fit "manualmente"; chiamando il metodo **Fit("pol1")** ed estrarre tramite il metodo **GetParameter(..)**.

La prossima volta vedremo alcuni comandi per migliorare l' "estetica " del grafico.....