Il C++ e la compilazione sotto Linux

Il C++

- Il linguaggio C viene creato negli anni 70 nei laboratori Bell
- Il linguaggio C e' stato concepito per essere vicino all'hardware e contemporaneamente offrire programmazione di alto livello
- Bjarne Stroustrup (Professore al Texas A&M University) sviluppa nel 97 il C++ come estensione del C con forte tipizzazione e supporto per una ampia tipologia di stili di programmazione:
  - data abstraction
  - object-oriented programming
  - generic programming
- *Nel corso di Fondamenti di Informatica I avete visto la programmazione procedurale*
- *Nel corso di Fondamenti di Informatica II vedrete tecniche di programmazione orientata agli oggetti*

Estensioni dei File

- Per convenzione i file utilizzati per la stesura di un programma C++ hanno le seguenti estensioni:
  - nome.cc
  - nome.cp
  - nome.cxx
  - nome.cpp
  - nome.c++
  - nome.C
- mentre i file di intestazione (header):
  - nome.h
  - nome.hpp

Linux

- Per poter scrivere un programma ed eseguirlo su un calcolatore occorre affidarsi ai servizi offerti da un sistema operativo (S.O.)
- I sistemi operativi di piu' comune utilizzo sono:
  - Windows
  - MacOS
  - Linux/Unix
- In questo corso noi faremo riferimento al S.O. Linux

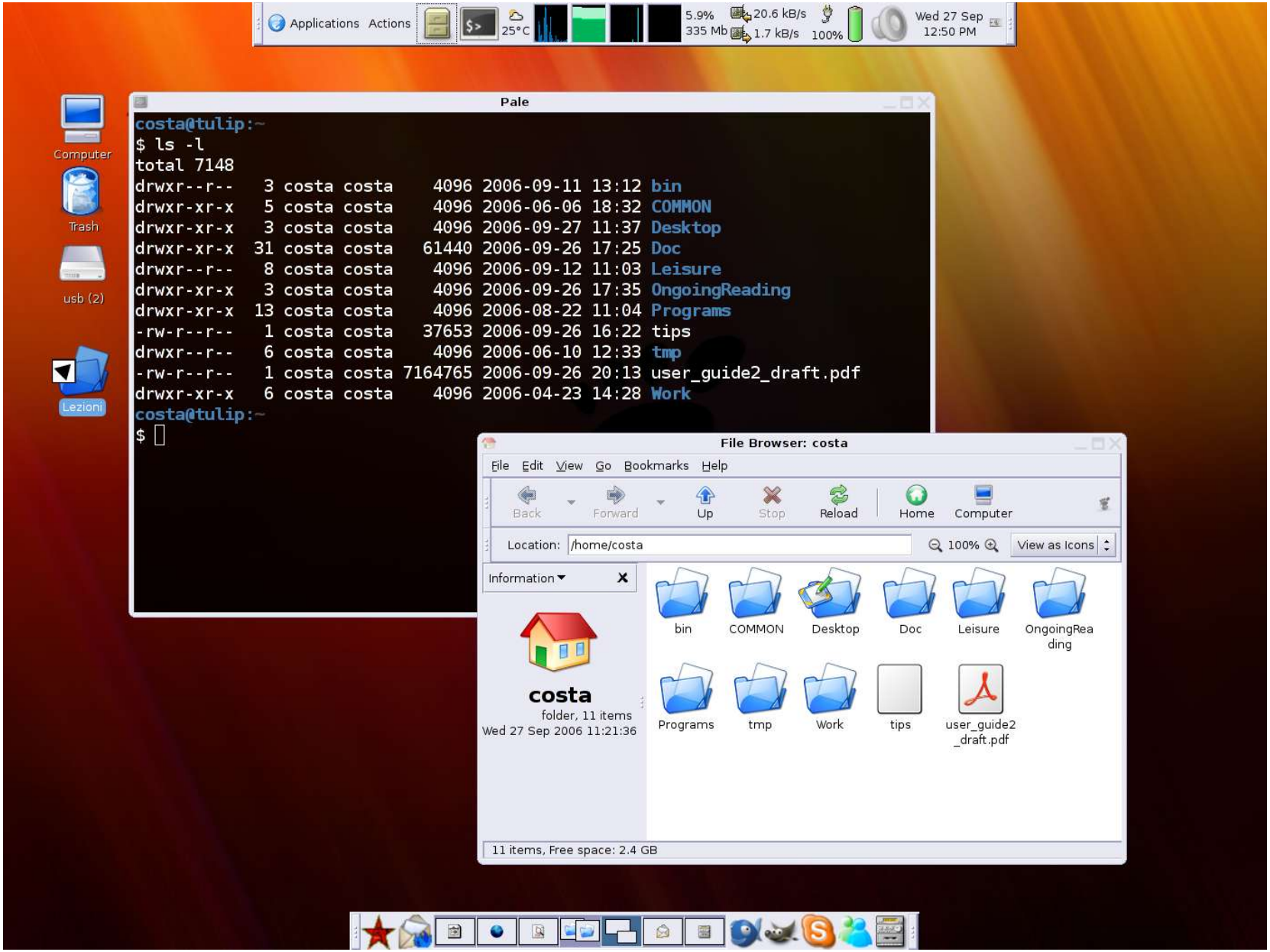
Linux in breve

- Per interagire con un S.O. sono generalmente possibili due modalita':
  - tramite una interfaccia grafica
  - tramite una consolle di comandi

- Nel primo caso ci si affida ad un sistema di puntamento (mouse) per selezionare delle icone che rappresentano dei programmi e si utilizzano dei menu per scegliere all'interno di questi quali azioni compiere
- Vantaggio: semplicita' ed intuitivita'

- Nel secondo caso si utilizza un programma chiamato interprete shell che rimane in ascolto di comandi digitati da tastiera
- Vantaggio: potenza espressiva

Esempio


Applications Actions
25°C
5.9%
335 Mb
20.6 kB/s
1.7 kB/s
100%
Wed 27 Sep
12:50 PM
Computer
Trash
usb (2)
Lezioni
Pale
costa@tulip:~
$ ls -l
total 7148
drwxr--r-- 3 costa costa 4096 2006-09-11 13:12 bin
drwxr-xr-x 5 costa costa 4096 2006-06-06 18:32 COMMON
drwxr-xr-x 3 costa costa 4096 2006-09-27 11:37 Desktop
drwxr-xr-x 31 costa costa 61440 2006-09-26 17:25 Doc
drwxr--r-- 8 costa costa 4096 2006-09-12 11:03 Leisure
drwxr-xr-x 3 costa costa 4096 2006-09-26 17:35 OngoingReading
drwxr-xr-x 13 costa costa 4096 2006-08-22 11:04 Programs
-rw-r--r-- 1 costa costa 37653 2006-09-26 16:22 tips
drwxr--r-- 6 costa costa 4096 2006-06-10 12:33 tmp
-rw-r--r-- 1 costa costa 7164765 2006-09-26 20:13 user_guide2_draft.pdf
drwxr-xr-x 6 costa costa 4096 2006-04-23 14:28 Work
costa@tulip:~
$
File Browser: costa
File Edit View Go Bookmarks Help
Back Forward Up Stop Reload Home Computer
Location: /home/costa
100%
View as Icons
Information
costa
folder, 11 items
Wed 27 Sep 2006 11:21:36
bin
COMMON
Desktop
Doc
Leisure
OngoingReading
Programs
tmp
Work
tips
user_guide2_draft.pdf
11 items, Free space: 2.4 GB

Scrittura di un programma in Linux

- Un programma sorgente e' un semplice file di testo
- Si utilizza un editor di testo per scrivere il programma
- Sotto Linux gli editor piu' comuni sono:
  - Emacs (Xemacs)
  - Pico
  - Vi (Vim)

- Esempio di programma C++

```
#include <iostream>
using namespace std;
int main() {
    cout << "Hello World with C++ streams!" << endl;
}
```

Compilazione in Linux

- Un programma sorgente viene tradotto in un linguaggio comprensibile al calcolatore da un programma chiamato compilatore
- Sutto linux il compilatore di riferimento e' GNU Compiler Collection (GCC)

- Per compilare si usa il comando

```
g++ <source>.cpp -o <executable binary>
```

- Per eseguire il programma compilato

```
./<executable binary>
```

Esempio


unix.cs.tamu.edu - PuTTY
> cat helloworld.cpp
#include <iostream>
using namespace std;
int main(int argc, char* argv[]) {
cout << "Hello World with C++ streams!" << endl;
}
> g++ helloworld.cpp -o helloworld
> ./helloworld
Hello World with C++ streams!
>

Compilazione di piu' file

- Quando si compila un programma costituito da piu' file sorgenti ci sono due possibilita':
  - compilare tutte le componenti e collegarle (link) insieme in un singolo passo
  - compilare separatamente le componenti e ottenere singoli file binari e successivamente collegarli insieme
- La prima opzione e' piu' semplice ma impiega piu' tempo
- La seconda opzione permette di compilare solo le parti che sono cambiate ed e' piu' veloce

Esempio

- Compilazione in un singolo passo:

```
g++ <source1>.cpp <source2>.cpp -o <executable binary>
```

- Compilazione in piu' passi:

```
g++ -c <source1>.cpp
g++ -c <source2>.cpp
g++ <source1>.o <source2>.o -o <executable binary>
```

Esecuzione [approfondimento]

- Per eseguire un programma in linux basta digitarne il nome

  `myprog`

- ..quando la directory in cui si trova il programma e' conosciuta dalla shell
- La shell conosce tutte le directory comprese in un elenco chiamato PATH che puo' essere modifica dall'utente
- Alternativamente si puo' indicare alla shell che si intende lavorare nella directory corrente
- La shell segue le seguenti convenzioni per indicare directory speciali:
  - ~ e' la home directory
  - . e' la directory corrente
  - .. e' la directory che contiene la directory corrente (padre)

Esecuzione [approfondimento]

- Per indicare una sotto-directory si separano i nomi con il carattere /

```
doc/lettere/lavoro
```

- per indicare un file in una directory si premette la directory e poi il nome del file

```
doc/lettere/lavoro/mario_rossi.txt
```

- per indicare un file nella directory corrente

```
./myprog
```

Permessi

- In linux ad ogni file e directory sono associate delle informazioni per permettere la gestione della sicurezza: proprietario (user) e gruppo di appartenenza (group)
- Quando si accede ad un sistema linux si acquisisce una identita' diventando un utente specifico
- Si puo' decidere di garantire l'accesso (read) o la possibilita' di modifica (write) o di esecuzione (execute) ad un file al solo proprietario, o ad un utente di un gruppo o a tutti gli utenti incondizionatamente
- Si modificano questi permessi con il comando

```
chmod u+rwx file
```

Note

- Quando il compilatore crea il programma oggetto per noi questo acquisisce i permessi corretti
- Tuttavia se qualcosa dovesse impedirlo e' possibile utilizzare il comando chmod per rendere il programma eseguibile

- Se il file viene spostato in un altro computer puo' perdere i permessi
- Se il file viene spostato in un altro computer per poter essere eseguibile deve avere:
  - lo stesso sistema operativo (per capire il formato oggetto)
  - la stessa architettura di CPU (per eseguire i comandi in binario)
  - lo stesso ambiente di esecuzione run time (per utilizzare le librerie dinamiche)

Avvertimenti in compilazione

- Il compilatore analizza il codice per verificare che sia sintatticamente corretto e solo dopo procede alla sua compilazione
- E' possibile pero' chiedere al compilatore di indicare quali operazioni sembrano sospette ancorche' sintatticamente corrette (warning)
- Istruzioni indicate come sospette possono essere errori logici o generare errori di compilazione su altri compilatori

```
g++ -Wall source.cc -o object.o
```

Debugging

- Quando si programma si vuole essere in grado di esaminarlo in fase di esecuzione (debugging)
  - controllare il valore delle variabili ad ogni passo di esecuzione
  - indicare dei punti di interesse (breakpoint) in cui fermare l'esecuzione
- Il programma eseguibile e' scritto in linguaggio macchina e pertanto risulta difficile per un programmatore seguire cosa avviene nel suo codice trasformato
- Per collegare le istruzioni di alto livello a quelle in linguaggio macchina e visualizzare solo quelle di alto livello durante la fase di debugging si indica al compilatore di inserire anche il codice di alto livello

```
g++ -g source.cc -o object.o
```

Ottimizzazione

- Quando il codice e' stato liberato da bug vogliamo compilarlo in modo che venga eseguito il piu' rapidamente possibile o in modo che sia il piu' compatto possibile
- E' possibile indicare al compilatore di eseguire delle ottimizzazioni in modo da ottenere codice con la stessa funzionalita' ma piu' rapido e compatto
- Maggiore e' il livello di ottimizzazione maggiore sara' il tempo necessario per la compilazione

```
g++ -O source.cc -o object.o
g++ -O3 source.cc -o object.o
```

La compilazione

- Il processo di compilazione si compone di diverse fasi
- Quando si invoca il compilatore con g++ in realta' stiamo invocando un programma che chiama in successione una serie di altri programmi fornendo a ciascuno il risultato dell'elaborazione del programma precedente
- Una tipica sequenza per la compilazione e':
  - pre-processor
  - compiler
  - optimizer
  - assembler
  - linker/loader
- Disponendo di diversi moduli si possono riutilizzare per diversi linguaggi (pre-processor e compiler) e diverse architetture (assembler e linker/loader)

Pre-processor

- **Pre-processor:** prende in ingresso un file .cc ed esegue le direttive come:
  - #include files: cioe' ricopia il contenuto dei file
  - #define macros: cioe' espande delle definizioni
  - #ifdef: cioe' inclusione condizionale del codice
- **Compiler:** traduce il codice C++ in assembler
- **Optimizer:** esegue ottimizzazioni su del codice che e' indipendente dal linguaggio (puo' essere utilizzato con piu' linguaggi)
- **Assembler:** traduce il codice assembler in codice macchina
- **Linker/loader:** unisce piu' codici oggetto in un unico eseguibile, in un formato gestito dal S.O. (la locazione del segmento dati, sorgente, debug info)