Costruzione di applicazioni

- Compilazione
- *Loading* (*linking*)
- Esecuzione

su *Linux*

Programmi semplici

- Per compilare e *linkare* programmi semplici:

  `c++ mytest.cpp`

  L'eseguibile sarà **`a.out`** (**`./a.out`**)
- Per cambiare il nome dell'eseguibile:

  `c++ -o mytest mytest.cpp`
- Per vedere quello che fa il compilatore:

  `c++ -v mytest.cpp`

Uso di librerie esterne

- Per compilare e *linkare* usando librerie esterne:

```
c++ -Iheader_dir -Llib_dir \
-llib_name mytest.cpp
```

- N.B. `lib_name` senza il prefisso **`lib`** e il suffisso **`.a`** o **`.so`**

Esempio: uso di CLHEP

- CLHEP è un pacchetto sviluppato a SLAC con classi utili per HEP
- Header files in: **`/cern/CLHEP/include`**
- Libreria (**`libclhep.a`**) in: **`/cern/CLHEP/lib`**

```
c++ -I/cern/CLHEP/include  \
 -L/cern/CLHEP/lib -lclhep \
 mytest.cpp
```

Programmi più complessi

- Se si hanno più files da compilare e linkare:
  - compilazione senza load dei singoli files:

```
c++ -c a.cc -o a.o
c++ -c b.cc -o b.o
```

  - load di tutti i files:

```
c++ mytest.cpp a.o b.o
```

Produzione di una libreria

- Meglio:
  - compilazione senza load dei singoli files:

```
c++ -fPIC -c a.cc -o a.o
c++ -fPIC -c b.cc -o b.o
```

  - creazione di una shared library:

```
c++ -shared -o libmylib.so \
a.o b.o
```

  - creazione di una archive library:

```
ar -r libmylib.a a.o b.o
```

Uso di *shared libraries*

- E’ necessario definire una variabile ambientale **`LD_LOAD_PATH`** che contenga il nome della directory in cui si trova la libreria prima di eseguire il programma
- Alternativamente si può compilare con **`C++`** aggiungendo l’opzione:

```
-Wl,-rpath lib_dir -Llib_dir\
-llibname
```

Uso di Makefile

- L’uso del comando **`make`** con un **`Makefile`** semplifica molto la vita in caso di pacchetti complessi!