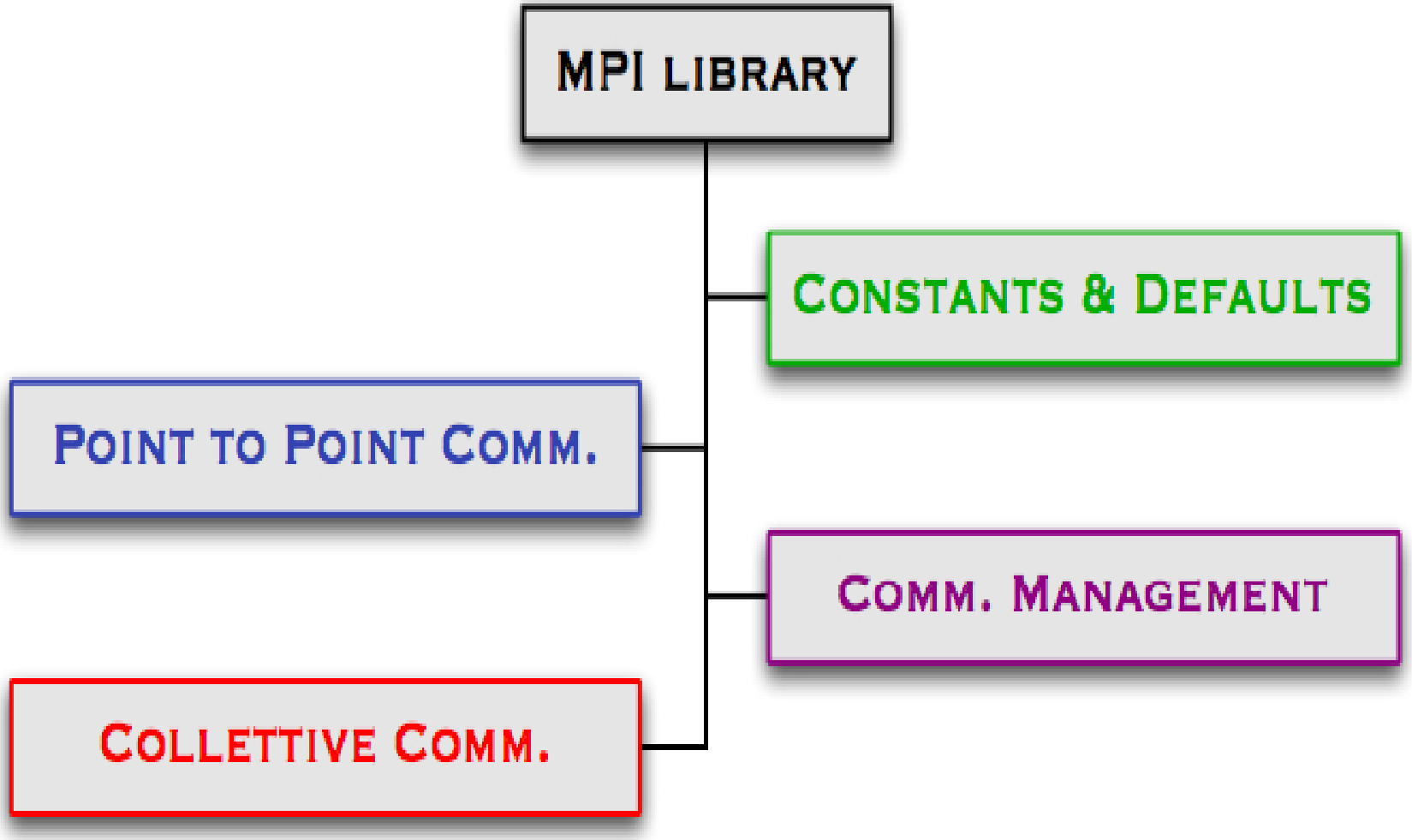
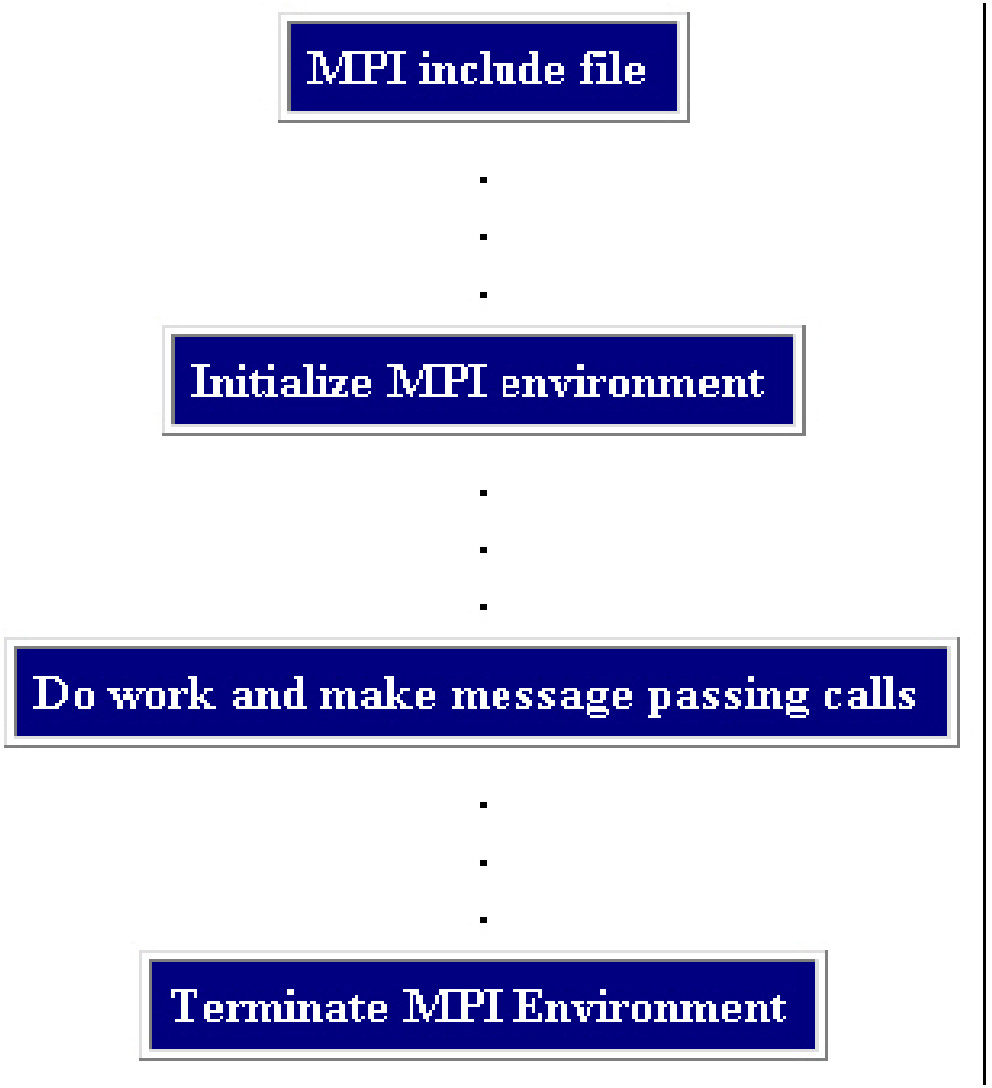
MPI – Formato delle chiamate

- In C: err = MPI_Xxxxx(parameter, …)
  - MPI_ e' prefisso di tutte le funzioni MPI
  - Dopo il prefisso, la prima lettera è maiuscola e tutte le altre minuscole
  - Praticamente tutte le funzioni MPI tornano un codice d'errore intero
  - Le costanti sono stringhe tutte in maiuscolo
- In Fortran (cenni sulle differenze tra Fortran e C): call MPI_XXXX(parameter,…, err)
  - MPI_ e' prefisso di tutte le subroutine MPI
  - Anche se il Fortran è case insensitive, le subroutine e le costanti MPI sono convenzionalmente scritte in maiuscolo
  - L'ultimo parametro è il codice d'errore (INTEGER)

MPI – Struttura della libreria



MPI – Struttura di un programma



MPI

Tutti i programmi MPI devono includere l'header file standard:
mpi.h per il C, mpif.h per il Fortran
Nell'include file standard sono contenute definizioni, macro e prototipi di funzioni necessarie per la compilazione di un programma MPI
(due parole a proposito dei file di header).

MPI_Init: La prima routine MPI in un programma MPI deve essere la routine MPI_Init:

- Inizializza l'ambiente di comunicazione
- All'interno di un programma MPI puo' essere chiamata una sola volta
- definisce il comunicatore standard MPI_COMM_WORLD

MPI_Finalize: In un programma MPI conclude la fase di comunicazione

- Provvede al rilascio pulito dell'ambiente di comunicazione
- Non e' possibile eseguire ulteriori chiamate MPI dopo la finalizzazione

Binding MPI_Init

In C:

```
int MPI_Init(int *argc, char **argv)
```

Se necessario, piu' essere utilizzata per passare gli argomenti di command line a tutti i processi

In Fortran

```
INTEGER err
MPI_INIT(err)
```

Binding MPI_Finalize

In C:

```
int MPI_Finalize(void)
```

In Fortran

```
    INTEGER err
MPI_FINALIZE(err)
```

MPI – Hello World!

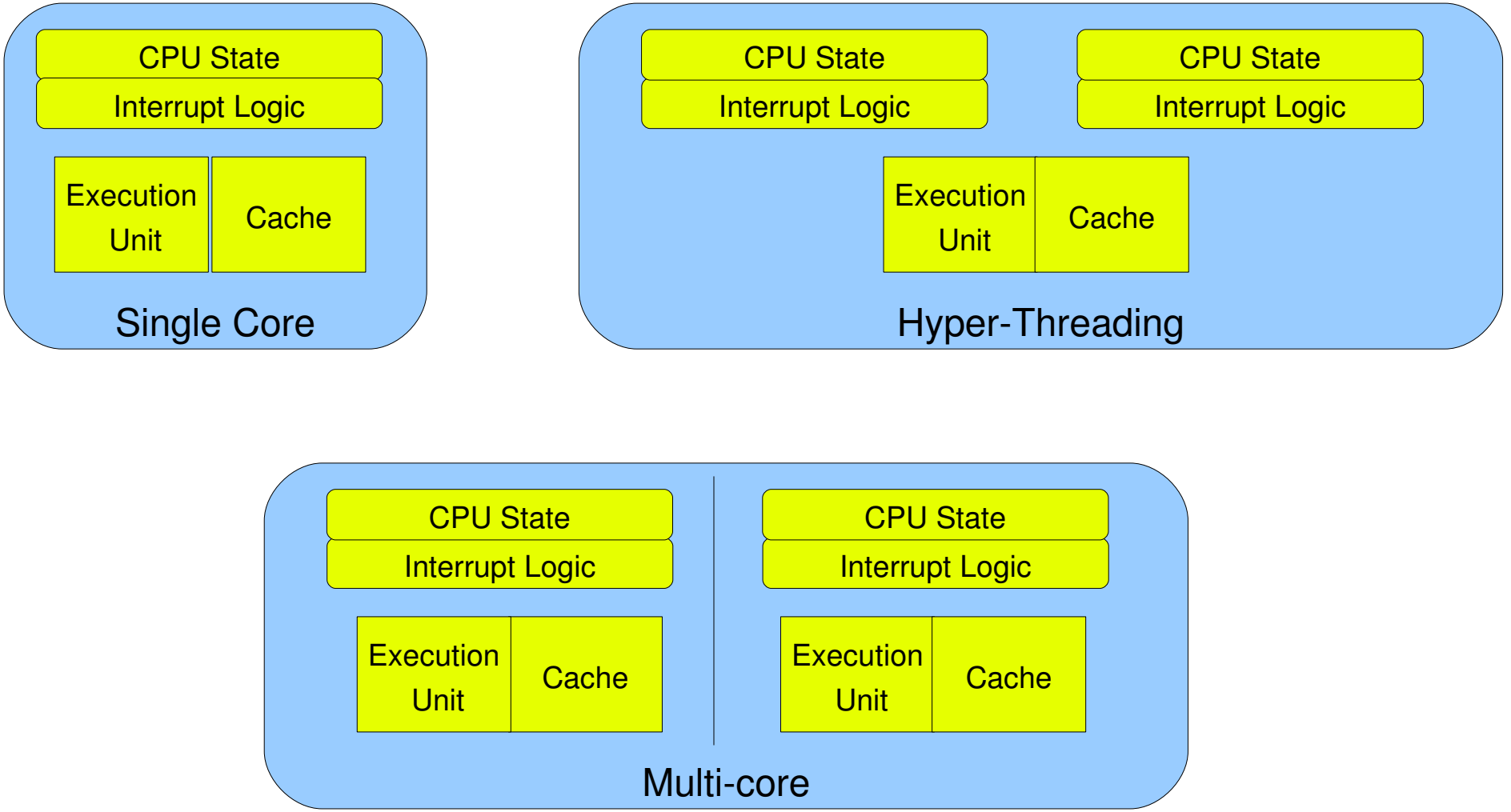
Vediamo il classico “Hello World!” usando MPI.
La compilazione e l'esecuzione di un programma MPI dipendono dalla particolare implmentazione che si sta usando. Nel nostro caso stiamo lavorando in un **“Cluster Beowulf Intel”**

- 8 nodes Intel Pentium 4 Hyper-Threading 3 Ghz
- Central Memory 1 GB on each node

Useremo mpich2-1.0.6 (compilato con GNU C)

- http://www-unix.mcs.anl.gov/mpi/mpich2/index.htm

Multi-core and Hyper-Theading



MPICH2

Come funziona ?

```
$ cat machinefiles
cg01
cg02
...

$ cat .mpd.conf
secretword=test

$ chmod 600 .mpd.conf

$ mpdboot -n 9 -f machinefiles -r rsh
$ mpdallexit

$ mpdlistjobs
$ mpdkilljob 11@cg00_34674
```

MPI – Hello World

Esempio: **hello.c**

```
$ mpicc -o hello hello.c
$ mpiexec -n 9 ./hello
```

vediamo adesso il kill: **tokill.c**

```
$ mpicc -o tokill tokill.c
$ mpiexec -n 4 ./tokill
$ mpdlistjobs
$ mpdkilljob 14@cg00_34674
```

(maggior dettaglio sul funzionamento di mpicc)