Robotica Mobile

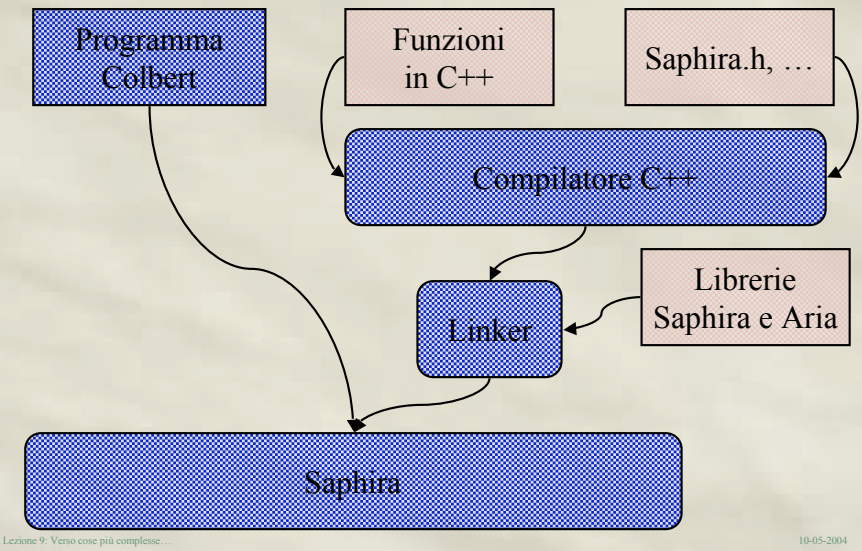
Lezione 9:
Verso cose più complesse…

10-05-2004

Activity labels

- `OnInterrupt:` this label is the target of the interrupt signal. When an interrupt signal is sent to the activity, it branches to this label, and executes any commands found after it. Usually these commands will halt the robot, or clean up in some way. Note that the activity only gets one cycle to perform these commands; any command that causes a wait in the Colbert executive (e.g., a `move()` command issued without `noblock`) suspends further processing, and the activity enters the interrupted state.
- `OnResume:` this label is the target for a resume signal sent to the activity. Whenever the activity is resumed from an interrupt or suspension, processing will start after this label. If the label doesn't exist, then processing on a resume starts from the first statement of the main body.

Un modo per programmare in Saphira



Avvertenze preliminari

- Le istruzioni saranno date solo per lavorare sotto LINUX: quelle per Windows sono contenute nella documentazione.
- Sotto Windows si tratta di creare delle DLL, usando Visual C++
- Le macchine del laboratorio hanno Mandrake 9.0 e quindi gcc-3.2, ma…
  - If you are compiling … Saphira programs, … you need to install gcc-2.95.3.
- Sulle macchine del laboratorio ci sono sia la versione 2.95.3 che la 3.2-1, con un patch come da file README.TXT. Comunque è meglio verificare:
  - gcc -v

Altre avvertenze

- È possibile personalizzare in vari modi la chiamata del simulatore. Per esempio:
  - export SAPHIRA=/usr/local/Saphira
  - alias pioneer="$SAPHIRA/bin/pioneer"
  - alias saphira="$SAPHIRA/bin/saphira"
  - alias robosim=pioneer -r $SAPHIRA/params/pion1m.p & saphira &"
  - alias robosim=pioneer -r $SAPHIRA/params/pion1m.p -w $SAPHIRA/worlds/ActivMediaLab.wld & saphira &"
- Non potete modificare le variabili di ambiente SAPHIRA e ARIA, ma potete cambiare secondo necessità LD_LIBRARY_PATH

La prima prova…

- La facciamo con la ben nota testload, spiegata in loadable.pdf
- I file relativi sono in $SAPHIRA/tutor/loadable, e sono:
  - testload.cpp
  - makefile
- Come già detto, il risultato della compilazione deve essere uno shared object e deve andare a finire in $SAPHIRA/lib
- testload.cpp lo abbiamo già esaminato, e comunque le istruzioni generali sono in colbert-user.pdf
- Diamo un'occhiata al makefile, che è più o meno lo stesso per tutti i programmi:

Il makefile (intestazione)

```
#############################################################

SRCD = ./
OBJD = ./
INCD = ../../ohandler/include/
BIND = ../../bin/
LIBD = ../../lib/
ARIAD = ../../../Aria/

# check which OS we have
include $(INCD)os.h

SHELL = /bin/sh

INCLUDE = -I$(INCD) -I$(ARIAD)include
```


Il makefile (parte esecutiva)

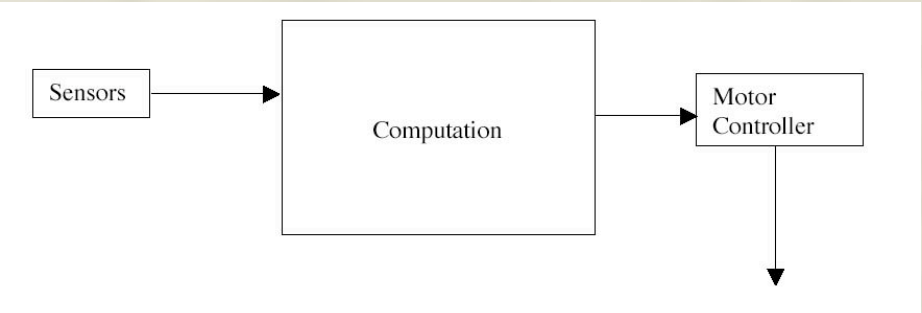
```
#############################################################
all: $(LIBD)testload.so
    touch all

$(OBJD)testload.o: $(SRCD)testload.cpp
    $(CC) $(CFLAGS) -c $(SRCD)testload.cpp $(INCLUDE) -o $(OBJD)testload.o

$(LIBD)testload.so: $(OBJD)testload.o
    $(LD) $(SHARED) -o $(LIBD)testload.so $(OBJD)testload.o
```

Riassumendo…

- Ciò che Colbert non offre, può essere scritto in C.
- In C, possiamo definire (e rendere disponibili in Colbert):
  - Costanti sfAddEvalConst
  - Variabili sfAddEvalVar
  - Funzioni sfAddEvalFn
  - Behavioral actions sfAddEvalAction
- Esaminare i file /usr/local/Saphira/tutor/crawl/init.act e /usr/local/Saphira/tutor/crawl/crawl.cpp
- C'è anche il makefile…
- Un esempio più semplice: in tutor c'è anche la directory swerve

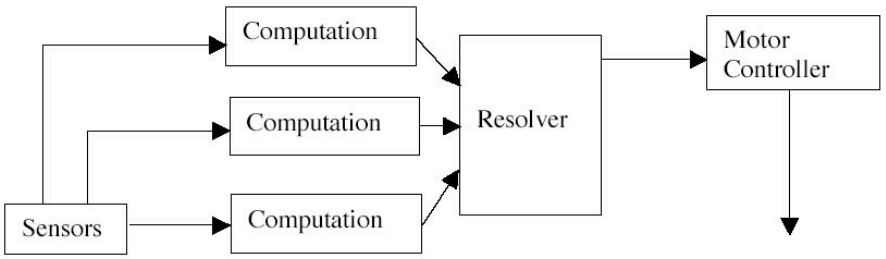
Le ragioni di questa complessità (1)

- Sistema di controllo tradizionale (monolitico):



Le ragioni di questa complessità (2)

- Controllo partizionato (indiretto):



E adesso passiamo alle prove pratiche!

- Provare a ricompilare uno shared object, dopo aver apportato qualche microscopica modifica
- Provare a scrivere uno shared object contenente funzioni che facciano qualcosa di sensato
- Studiare bene gli esempi, e provare a scrivere qualche behavioral action (modificando quelle esistenti, mi racomando!)
- Le prossime lezioni riguarderanno la fuzzy logic, e spiegheranno bene come scrivere le behavioral actions.