Epics Server – Guida interna WP09

In questa breve guida illustrerò quello che ho imparato installando un Server Epics C++ di esempio.

Il primo step è quello di scaricare il pacchetto **Epics Base**: è una cartella compressa che si può trovare cercando sulla pagina dedicata a cui si può arrivare digitando su Google "Epics Base" (il link, se non è cambiato, potrebbe essere http://www.aps.anl.gov/epics/base/). Si consiglia di scaricare l'ultima versione disponibile.

Dopo aver scaricato il pacchetto, si può procedere con la sua decompressione.
Dopo averlo decompresso si deve, da terminale, entrata nella cartella appena creata.
Prima di poter digitare il comando *make* occorre necessariamente impostare la variabile d'ambiente **EPICS_HOST_ARCH**. Per farlo bisogna navigare nella cartella **startup** ed eseguire lo script *"EpicsHostArch"*. Lo script stamperà il valore che deve essere assegnato alla variabile **EPICS_HOST_ARCH**. Per impostarla, dunque, è sufficiente digitare:

export EPICS_HOST_ARCH="linux-x86_64"

linux-x86_64 era ciò che ha stampato lo script *"EpicsHostArch"* sulla mia architettura.

Dopo aver settato la variabile, si può procedere alla compilazione del tutto digitando *make*. È possibile che la compilazione non vada a buon fine a causa della mancanza della libreria **readline.h** o di altre librerie di sistema. Per ovviare al fatto si deve ricercare su Google il modo di installare sul sistema quelle librerie.

Una volta compilato il pacchetto, nella cartella **bin/nome_architettura**, si possono trovare programmi molto interessanti, quali:

softIoc: lancia una *ioc shell* (il diminutivo è *iocsh*), un ambiente che presenta alcune funzionalità della shell del sistema operativo *real-time* ***vxWorks***.
Con questo strumento è possibile caricare database di variabili PV e renderle disponibili ai programmi clienti.
Per farlo basta caricare i database con i comandi appositi e digitare *iocInit* (N.B. digitare *iocInit* dopo aver caricato un database è fondamentale per rendere disponibili le variabili ai programmi clienti!).
In particolare, supponendo che il file **test.db** contenga dei records di esempio, è possibile lanciare il programma digitando ***./softIoc -d test.db*** (esso inizializza il database presente in **test.db** e lancia in automatico il comando *iocInit*). Per controllare i records caricati si può digitare *dbl*. Le funzionalità offerta da una *iocsh* sono notevolmente documentate (è anche possibile lanciare una di queste shell o far eseguire un insieme di comandi da un programma C con le funzioni *iocsh* o *iocshCmd*).
Una volta lanciato il programma ***./softIoc -d test.db*** è possibile monitorare le variabili PV così create con i programmi che decriverò in seguito.

Riporto il contenuto del file **test.db**:

```
# EPICS database to use while testing and developing pvMail.py code

#  /APSshare/epics/base-3.14.12.1/bin/linux-x86-el5-debug/softIoc -d test.db
#
#  IOC:     softIoc -d test.db
```

```
#  client:  camonitor pvMail:{trigger,message}
#  pvMail:  pvMail.py  pvMail:trigger pvMail:message prjemian@gmail.com,jemian@anl.gov

record(bo, "pvMail:trigger")
{
        field(DESC, "trigger PV")
        field(ZNAM, "off")
        field(ONAM, "on")
}
record(stringout, "pvMail:message")
{
        field(DESC, "message to be sent by email")
        field(VAL,  "Mio messaggio 3")
}

# Copyright (c) 2014, UChicago Argonne, LLC.  See LICENSE file.
```

Prima di procedere alla decrizione degli altri programmi mi soffermo sulla descrizione di una particolarità dei software **Epics**: le variabili d'ambiente (*Environment variables*).
Infatti, ogni programma e ogni libreria del pacchetto **Epics Base** usa le variabili di ambiente (la scelta è fatta per far si che i programmi non tengano traccia di alcuni informazioni importanti usando, ad esempio, un file di configurazione, ma le vadano a cercare in queste variabili d'ambiente).

Le più rilevanti ai nostri fini sono (oltre alla già citata **EPICS_HOST_ARCH**):

export EPICS_CA_SERVER_PORT=5094 → La porta dove il client Epics va a cercare una PV.

export EPICS_CAS_SERVER_PORT=5094 → La porta dove il server Epics va a cercare una PV

export EPICS_CA_AUTO_ADDR_LIST=NO → Indica a Epics di non cercare le PV in una lista di indirizzi generata in automatico.

export EPICS_CA_ADDR_LIST=192.84.144.161 → Lista di indirizzi IP dove il client Epics va a cercare una PV.

export EPICS_CAS_INTF_ADDR_LIST=192.84.144.161 → Lista di indirizzi IP dove il server Epics mette a disposizione le PV.

*N.B. Prima di ogni variabile ho riportato anche il comando export che serve per impostarla nella shell. C'è da ricordare anche che se si ha intenzione di scrivere uno script che le imposti in automatico, esso deve **necessariamente** essere lanciato con il comando **source**.*

Sottolineo che il programma **softIoc** fa già da server **Epics** (tanto che con il comando *casr* è possibile ottenere addirittura la versione usata del **Channel Access Server**).

Il programma **softIoc** non fa altro che mettere a disposizione una *ioc shell*, quindi un programma C che ne crei una con la funzione *iocsh*, o che lanci comandi con la funzione *iocshCmd*, ha le stesse potenzialità del programma **softIoc** (compreso il lato server!).

Sottolineo che le variabili d'ambiente devono essere settate ogni volta che si crea una nuova **shell**. Per stamparne il valore si può usare il comando **echo $nome_variabile**.
Esse possono essere settate anche via ioc shell con il comando apposito (dovrebbe essere setEnv(...) ma non ne sono sicuro, occorre controllare la documentazione).

cainfo: il programma si usa digitando **./cainfo PVName** (ad esempio, se si è caricato il test.db, ./cainfo pvMail:message). Esso stampa le informazioni sul Channel Access associata alla variabile PV di nome PVName. Può essere utile quando per fare dei test di connessione quando si modificano le variabili d'ambiente riportate sopra.

camonitor: si usa digitando **./camonitor PVName**. Il programma serve per monitorare la variabile PV di nome PVName.

caget: si usa digitando **./caget PVName**. Il programma serve per ottenere il valore della varibile PV di nome PVName.

excas: lancia un server Epics. Questo serve mette a disposizione una lista predefinita di varibili PV (ad esempio “jane” o “fred”). Esse possono essere controllate con i programmi visti sopra (ad esempio **./camonitor “jane”**).

Compilare i pochissimi esempi di sorgenti Epics scritti in C++ che si trovano in rete è molto complicato. Per fortuna, il pacchetto Epics Base offre un meccanismo per ottenere dei sorgenti senza errori con tanto di Makefile.

Per usare questo meccanismo occorre individuare lo script Perl “*makeBaseApp.pl*” (tendenzialmente esso è nella cartella dove sono presenti anche i programmi descritti prima).
Poi, occorre creare una cartella con un nome a piacere e digitare, all'interno della cartella, il seguente comando:

path_dello_script_makeBaseApp.pl/makeBaseApp.pl -t caClient caClient

I vari modi per utilizzare lo script *makeBaseApp.pl* sono descritti in rete.

Nel nostro caso, possiamo digitare:

path_dello_script_makeBaseApp.pl/makeBaseApp.pl -t caClient caClient

per ottenere l'applicazione client,

path_dello_script_makeBaseApp.pl/makeBaseApp.pl -t caServer caServer

per ottenere l'applicazione server,

path_dello_script_makeBaseApp.pl/makeBaseApp.pl -t ioc ioc

per ottenere un semplice programma C che lancia una **ioc Shell**.

Dopo aver eseguito un comando di questi, appare, nella cartella dove lo abbiamo digitato, un'altra cartella contenente il programma richiesto.

Ad esempio, digitando nella cartella **/home/daniele/Scrivania/Esempi**

il comando:

path_dello_script_makeBaseApp.pl/makeBaseApp.pl -t caServer caServer

otteniamo la cartella:

/home/daniele/Scrivania/Esempi/caServerApp

essa contiene i sorgenti C++ di una applicazione server e il Makefile.

Per compilare i sorgenti è sufficiente digitare il comando *"make"*.

L'eseguibile ***casexample*** generato nella cartella **O.tipo_architettura** funziona in modo analogo al programma ***excas*** descritto prima. Quindi, basterà impostare le variabili d'ambiente, lanciare il server e monitorare le variabili predefinite con i programmi client descritti sopra.

*Occorre tener presente che l'applicazione server di esempio usa la libreria di funzioni denominata **Channel Access Portable Server**. Quindi, per capire il funzionamento dell'applicazione server esemplificativa, occorre documentarsi su questa libreria di funzioni.*

Il passo successivo sarà integrare il codice C++ del server con i nostri programmi.

*N.B. Occorre ricordarsi, usando gli strumenti presentati in questo documento, di impostare correttamente le variabili d'ambiente sopra descritte (si faccia particolare attenzione alla variabile **EPICS_HOST_ARCH**).*