Joint Project制御システム開発環境の標準化のための覚書 その1

加速器研究施設
山本　昇
Accelerator Lab.,
KEK, Tsukuba, JAPAN, 305-0801

original March 17, 2002
reviced on April 23, 2002

Abstract

アプリケーション開発環境の構築の第一段階として、EPICS Tool kit 上に制御システムを構築する際のディレクトリ構造について考察する。

1　加速器運転時に必要とされるソフトウェア等

ジョイントプロジェクトの加速器制御システムのアプリケーション開発環境の標準化を進めるにあたって、まず始めに、CVS repository や開発環境の directory 構造をある程度決めておくことが必要である。開発環境のディレクトリ構造は結局の所運転時のソフトゥエアのディレクトリ構造の写しであるので、まずは運転環境のディレクトリ構造の方針を決めておく必要があるだろう。制御システムが運転状態になった時に必要とされるプログラムや設定データなどを列挙してみると、表-1 にあげたようになった。

制御アプリケーションの幾つかは、夫々の初期設定ファイルが必要である。また、ビーム軌道制御プログラム等では、軌道調整用電磁石の値の組を一つの設定ファイルとして保存することが必要である。これらの各種の設定ファイルは、現在の設定ファイルとそれらの設定ファイルの変更毎に記録として残されるものの二種類がある。

KEKB の場合を参考に必要なデータを表-1 にあげてみた。

現在利用されている設定ファイルは、システムで一つの場所にまとめて置いておく。

また、これらの設定ファイルを変更した場合には運転用ファイルは一定の名前をもったリンクとし、本体のファイルは日付を含むファイル名をもつファイルとする。記録ファイルは必要に応じて、日毎、月ごと、年ごとなどのディレクトリを作成して保存する。といったルール作りが必要である。変更記録ファイルは現在の設定値ファイルとは別ディレクトリに整理する。

あるいは設定ファイル自体が ASCII のファイルであればこれを CVS 管理することも可能である。

軌道などの測定データは CVS 管理にはなじまないとおもわれる（更新頻度がかなり多い）

2　運転用ディレクトリ

前節であげた運転時に必要とされるソフトウェアやデータは、EPICS, GNU, Tornado, ORACLE など 特定の加速器には依存しないツール群と制御対象とな加速器固有のアプリケーションおよびデータに分けられる。特定の加速器に必要なアプリケーションに関係するソフトウェア・データについてのディレクトリ構造は表-2 のようなものになるだろう。[1]運転システムとしてはこの他に EPICS や GNU ソフトウェアなどのツール群、商用ソフトウェアなどが必要であるが、これらを含めたディレクトリ構造は、後程 (第 5 節) で再び議論する。

device_app/, config/, operation_application/については CVS 管理を行う事が必要であろう。

[1] このツリー図は実際のディレクトリ構造そのもので無くてもよいが、これらの機能を持ったディレクトリがシステムの何処かに存在するはずである。

Part 1 Tools

1. Unix standard Tools
   (a) /usr/bin など
2. GNU などの tool
   (a) /jk/local/{etc, bin, lib, src}
3. EPICS tools
   (a) 通常のユーザが利用するディレクトリとしては/jk/local/epics/{etc, bin, lib} を用意。これらのディレクトリは自動的に、$(EPICS)/base/bin/<arch>、$(EPICS)/extensions/bin/<arch> などにマップされる仕組みを用意する。
4. vxWorks Kernel
   (a) $(TORNADO)/target/<arch>
5. 3rd party (commercial) tools:ORACLE, CAPFAST, など Tornado/VxWorks もこれと同様に扱うことも出来よう。
   (a) /opt/<products>
6. その他のアプリケーション、例えば SAD など。
7. SAD : binary file と Packages ディレクトリ, 環境変数
8. Python: binary と Library, 環境変数

Part 2 制御 Applications

1. デバイス/IOC Applications
   (a) IOC Applications
   (b) startup command
   (c) database
   (d) appLib
   (e) display
   (f) HOST side application
2. SAD script, library
3. Python script, library
4. 制御パネル定義ファイル
5. その他のアプリケーション

表- 1: 加速器制御に係わるソフトウェア等

1. Optics・Lattice
   (a) SAD input data
   (b) Official Lattice/Optics に基づく Optics の標準値？
2. Steering
   (a) Standard Setting
   (b) 変更記録
   (c) 軌道測定データ
   (d) 基準軌道 (Golden Orbit) とその変遷。Current の Golden Orbit は Link とする。本体は日付付きのファイル名
   (e) 軌道測定データ記録
3. 動的データのログ (kblogrd や archiver)
4. Bumpless restart 用のログ (casave)
5. 様々な運転アプリケーションの画面スナップショットなど。
6. 運転記録
7. 運転ログノート

表- 2: 運転上必要となる各種設定データなど

```
$(JKOPROOT) ┬ CVS/ :CVS 管理用
            ├ config/┬ ar/ :Archiver 設定など
            │        ├ restart/ :Bumpless Restart 設定など
            │        └ etc/ :環境変数設定スクリプトなど
            ├ device_app/
            └ operation_app/

$(JKLOGROOT) ┬ bootlog/     :IOC ブート時刻の記録
             ├ ioclog/        :IOC 運転時のコンソール出力の記録
             ├ orbit/        :オービットデータ
             ├ Archiver/     :アーカイバのログデータ
             └ restart/      :casave 起動設定ファイルなど
```

表- 3: 運転用環境のディレクトリツリー

```
$(CVSROOT) ┬ CVSROOT/          :CVS 管理用ファイル
           ├ config/           :制御システム各種設定ファイル
           ├ device_app/       :device/IOC アプリケーション　作業ディレクトリ
           ├ operation_app/    :ホストアプリケーション　作業ディレクトリ
           └ ....             :その他 CVS で管理する必要のあるもの。(paper, report の原稿など?)
```

表- 4: CVS repository のディレクトリツリー

```
$(JKDEVROOT) ┬ CVS/                  :CVS 管理用ファイル
             ├ config/┬ ar/       :Archiver 設定など
             │        ├ restart/  :Bumpless Restart 設定など
             │        └ etc/      :環境変数設定スクリプトなど
             ├ device_app/          :device/IOC アプリケーション　作業ディレクトリ
             └ operation_app/       :運転用ホストアプリケーション　作業ディレクトリ
```

表- 5: 開発作業用ディレクトリ

Logged Data/Archived Data はデータ量が膨大であること、バイナリデータも存在するであろうことを考えて、CVS では管理しない。ただし、データファイル名の生成には規則を設けておくことが必要である。ルールに基づくファイル名を生成する API を用意して置くことが望ましい。

アプリケーションの設定ファイル等は、$(JKOPROOT)/config に置く。それら設定ファイルの変更記録は、$(JKLOGROOT) の下に置く。$(JKLOGROOT) の本体はオペレーション計算機とは別のファイルサーバ計算機の上にあることになるだろう。

3 CVS repository

CVS は下図の様に operation ディレクトリで CVS 管理されるべきディレクトリ (/device_app, /config, /operation_app) と同様の構造を持つ。

運転用ディレクトリ、開発用ディレクトリはいずれもこの CVS repository から check out した Working ディレクトリとして作成される。
[2]

4 開発用ディレクトリ

開発用ディレクトリは基本的には運転用ディレクトリと同様であるが、data storage に関する部分は不要（？）である。

5 ルートディレクトリ

これらを総合すると制御システムの root には次の図のようなディレクトリが存在する。dev_app ディレクトリには各システム固有のアプリケーション (デバイスサポート、シークエンサ・プログラムなど〕が作成される。
[3]

[2]CVS ではさらに、report や paper の原稿、Web page なども管理することになる可能性がある。それらと制御ソフトウェアのツリーを如何に分離するか？

[3]Commercial software も異なるバージョンのものが共存する可能性、例えば Tornado 1 と Tornado 2, がある。このような場合はどうするのか？KEKB では正直言って場当たり的にインストールしてきた。(ディスク容量の制限という側面もあったにはあったが。)

```
$(JKROOT) ┬ opr/          :$(JKOPROOT) , 運転環境ディレクトリ
          ├ log/          :$(JKLOGROOT), 運転データなどの記録
          ├ dev/          :$(JKDEVROOT), 開発環境ディレクトリ
          ├ epics/        :$(JKEPICS), EPICS ディレクトリ
          │     ├  R3.13.4/
          │     │     ├ base/
          │     │     ├ extensions/
          │     │     ├ lib/<arch>/
          │     │     └ bin/<arch>/
          │     │
          │     ├  R3.13.6/
          │     │     ├ base/
          │     │     ├ extensions/
          │     │     ├ lib/<arch>/
          │     │     └ bin/<arch>/
          │     │
          │     ├  R3.14.1/
          │           ├ base/
          │           ├ extensions/
          │           ├ lib/<arch>/
          │           └ bin/<arch>/
          └local/              : $(JKLOCAL)/, Free software etc.
                ├ lib/         : link to $(JKMISC)/<arch>/lib/
                ├ bin/         : link to $(JKMISC)/<arch>/bin/
                ├ share/
                │    └ Packages   :SAD 標準
                │
                └ src/         :source code
                     ├ GNU/
                     ├ SAD/    :SAD source code
                     ├ Python/
                     └ PostgreSQL/

$(JKMISC)/
        ├<arch>/        : <arch> = hp700 | Linux | ....
        │      ├ bin/   :  pointed by $(JKLOCAL)/bin
        │      ├ lib/   :  pointed by $(JKLOCAL)/lib
        ...

$(JKCVSROOT)/      :CVS リポジトリ

/opt/               :Commercial procucts
   ├ CAPFAST/      :CAPFAST ディレクトリ $(JKCAPFAST)
   ├ Tornado/      :Tornado ディレクトリ $(JKTORNADO)
   └ ORACLE/       :ORACLE ディレクトリ  $(JKCAPFAST)

$(JKWORK)/         :作業用一時ファイルのためのディレクトリ
```

表- 6: ジョイントプロジェクト加速器制御システム関連のディレクトリ一覧

5.1 アプリケーションディレクトリの階層化

運転環境ディレクトリおよび環境開発ディレクトリの双方に、operation_app/ディレクトリと dev_app/ディレクトリが存在する。これらのディレクトリは CVS リポジトリからチェックアウトされ運転環境と環境開発で同じ構造を持つ。dev_app/は基本的に IOC レベルのアプリケーション、すなわち夫々のデバイスを制御するための Device support や Sequencer program, 必要な場合には固有のレコードサポート等を含む。また、夫々の機器を制御・テストするための操作パネル定義や機器固有のホスト側制御アプリケーション（SAD/Python scripts) 等を含む。

夫々のサブグループに付いて、これらの開発・運転のためのディレクトリは EPICS の標準配布ソフトウェアに含まれている makebaseApp コマンドを利用する。makebaseApp が使うテンプレートを拡張することで、display/scripts/documents 等のディレクトリが自動的に作成されるようにする。[4]

サブシステムはさらに階層構造を持つ可能性がある。どこまで階層を許すかは運用上の問題となる。サブ階層を許したときには、そのレベルでの一貫性を保つことが必要。

制御アプリケーションのためのディレクトリ構造としては、古川氏の CVS 文書にあるように、高位のレベルで、加速器のセクション分けを行い、夫々のセクションで IOC 関係のソフト、運転用アプリケションなどを夫々のセクション毎に管理する方法も考えられる [2]。

> 管理を容易にするために、Repository の First level（及び Second Level） directory は管理者だけが作れるようにし、
> 例えば、
>
> ```
> First Level (Group 単位):
>   Control Epics Linac RCS MR BT BL
>
> Second Level (Application 単位):
>   ioc-app  beam-app  misc-app  config
> ```
>
> などを作っておき、個々 Project はその下に作るのがよいのではないかと思 われる。（計算機上での加速器の名前、Linac 等、も早く決める必要がある。）

この方法の一つのメリットは、ioc アプリケーション作成者と運転アプリケーション作成者が同じ場合には、関係したディレクトリが一塊になっており、分かりやすいということがある。ただし、KEKB のように IOC アプリケーション開発者と、運転用アプリケーション開発者が異なることが多い場合には、そのメリットの意義はそれほど多くないのではないかと考える。また、この場合には複数のセクションにまたがるアプリケーションの取り扱いの約束が必要である。

標準の makeBaseApp.pl では, makBaseApp.pl コマンド実行時に、

```
 config
 iocBoot
xxxApp/Db/
xxxApp/src/
xxxApp/Makefile
```

が作られる。template に

```
xxxApp/display/
xxxApp/documents/
xxxApp/scripts/
```

などのディレクトリやサンプルファイルを追加しておくことで、adl ファイルや SAD/Tcl/Python などのスクリプトによるアプリケーションを入れる場所を誘導することが出来るだろう。[5]

[4] これらのディレクトリは加速器 - ハードウェアグループ - ソフトウェアコンポーネントと言う階層で様々なものを分けている。逆にソフトウェアコンポーネント - 加速器 - ハードウェアグループ という階層で物事を検索・整理出来る方法 (cross reference) が必要であるとの指摘があった。特に display や document ではこれらを root とする階層構造で検索・整理が必要な場合、例えば Web Page への出版、もある。

[5] 各ディレクトリに.description ファイルをおいてはどうだろうか？（ファイルシステムをオブジェクト・ストアシステムと見なせば、こういった考え方に正当性をあたえることも出来よう。）

```
operation_app/ ┬ Operation/
               │        ├ document :
               │        ├ display : display file (.adl, ...)
               │        ├ script :
               │        ├ lib :
               │        │    ├python : python library (.py, .pyc, .pyo)
               │        │
               │        ├ Packages : SAD パッケージ (library, .n)
               │
               ├ Control/
               ├ Linac/
               ├ BT/ or L3BT/
               ├ 3GeV/ (RCS/)
               ├......

device_app/ ┬ Control/
            ├ Linac/
            ├ BT/ or L3BT/
            ├ RCS/
            │    ├ RF/
            │    │  ├ iocBoot/
            │    │  ├ config/
            │    │  ├ KlyApp/
            │    │   ...
            │    ├ MG
            │    │  ├ iocBoot/
            │    │  ...
            │    ├ VAC/
            │    ├ Monitor/
            │    ├ Control/
            │    ├ Timing/
            │    └ Safety/
            ├ MR/
                  ├ MG/
                  ...
```

表- 7: IOC アプリケーションディレクトリ

```
<hw group>/ ┬ config/
            ├ Makefile
            ├ iocBoot/
            │       ├ iocxxx
            │       ├ iocyyy
            │         ....
            ├ xxxApp/
            │     ├ src
            │     ├ Db
            │     ├ display/  :device sepcific な 操作・パネル定義
            │     ├ scripts/  :device sepcific な Host アプリケーション
            │     ├ documents/ :各デバイスおよびアプリケーションについての説明
            │       ...
            ├ yyyApp
            ....
```

表- 8: makeBaseApp の作るディレクトリ

6 標準環境ディレクトリの実装

これまでの表に現れたディレクトリを表す環境変数は、JAERI-KEK Joint Project では、次の表 (表-9) のように定義される。

7 アプリケーション毎の環境変数

EPICS やその他のツールは夫々の環境を定義するための環境変数を持っている。ここでは EPICS を基本とする制御システムで必要になるであろう環境変数をまとめておく。

7.1 EPICS

EPICS の使用する環境変数は表-10 の通り。

7.2 Tornado

Tornado/VxWorks に関する環境変数などは EPICS/makeBaseApp 環境下で作業を行っている限りでは、base/config/CONFIG_SITE に必要な設定をして置けばよい。EPICS/makeBaseApp 以外の環境の下で VxWorks プログラムをコンパイルするには、表-11 の環境変数を適切に設定する。

7.3 ORACLE

8 generating setup file

8.1 SAD

SAD が使用する環境変数を表-13 に示す。

8.2 Python

Python が使用する環境変数を表-14 に示す。

環境変数名	ディレクトリ
CVSROOT	$(JKCVSROOT)
JKCVSROOT	/jkcvs
JKROOT	/jk
JKOPROOT	$(JKROOT)/opr
JKLOGROOT	$(JKROOT)/log
JKDEVROOT	$(JKROOT)/dev
JKCAPFAST	/opt/CAPFAST
JKTORNADO	/opt/Tornado
JKCAPFAST	/opt/ORACLE
JKLOCAL	$(JKROOT)/local
JKEPICS	$(JKROOT)/epics
JKMISC	/jkmisc
JKWORK	/scratch

表- 9: 標準環境の JKJP での実装

環境変数名	使用法・標準的な値
EPICS	$(JKEPICS)
HOSTARCH	ホストの機種 [hp700|alpha|solaris] 等
EPICS_CA_AUTO_ADDR_LIST	NO
EPICS_CA_ADDR_LIST	
EPICS_CA_SEARCH_ADDR_LIST	$(EPICS_CA_ADD_LIST)
EPICS_AR_PORT	7002
EPICS_IOC_LOG_PORT	7004
EPICS_TS_MIN_WEST	-540
EPICS_IOC_LOG_INET	iocLogServer の IP アドレス
EPICS_IOC_LOG_FILE_LIMIT	100000
EPICS_IOC_LOG_FILE_NAME	/tmp/iocLog
EPICS_DISPLAY_PATH	MEDM 等の adl ファイルの検索パス
PSPRINTER	既定の出力プリンタ
DM2K_COLOR_RULE	既定のカラー・ルール・ファイル
DM2K_GRAPHIC_RULE	既定のグラフィック・ルール・ファイル

表- 10: EPICS 環境変数

WIND_BASE	/cont/tornadoPPC
WIND_HOST_TYPE	parisc-hpux10
WIND_REGISTRY	abco1
WIND_LMHOST	abco1
PATH	$PATH:$WIND_BASE/host/$WIND_HOST_TYPE/bin

表- 11: Tornado 環境変数

環境変数名	使用法・標準的な値
ORACLE_BASE	/opt1/oracle
ORACLE_HOME	$ORACLE_BASE/product/7.3.2
ORACLE_SID	KEKB
ORACLE_TERM	xterm
ORACLE_DOC	$ORACLE_BASE/doc
ORA_NLS32	$ORACLE_HOME/ocommon/nls/admin/data
path	($ORACLE_HOME/bin . $path /usr/sbin $ORACLE_HOME/bin)
NLS_LANG	Japanese_Japan.JA16EUC

表- 12: コマンド・ライブラリ　パス

環境変数名	使用法・標準的な値
SAD$PACKAGES	/jk/local/share/Packages

表- 13: SAD 環境変数

環境変数名	使用法・標準的な値
PYTHONPATH	コロン':' で区切られたモジュールの検索パス
PYTHONSTARTUP	定義されていれば, 対話モードの起動時に実行される。
PYTHONHOME	定義されていれば, Python がインストールされた場所を示す。
PYTHONINSPECT	定義されていれば, -i オプションと同じ。
PYTHONUNBUFFERED	定義されていれば, -u オプションと同じ。stdin,stdout を unbuffered のバイナリモードで開く。

表- 14: Python 環境変数

CVSROOT	CVS repository のパス名
CVSREAD	設定されていれば、check out 時等にファイルを read only にする。
RCSBIN	ci,co などの RCS コマンドへの完全パス名。
CVSEDITOR	ログを記録する際に使用されるエディタ
CVSIGNORE	ci/co の対象外とするファイル名のパターンのリスト
CVS_IGNORE_REMOTE_ROOT	設定されていると、CVS/Root ファイル中のリモート・リポジトリを無視する。
CVS_RSH	ssh
CVSBIN	CVS binary installed location.(optional)
CVSLIB	CVS library installed location.(optional)

表- 15: CVS 環境変数

PATH	コマンド　パス
LD_RUN_PATH	実行時の動的ライブラリパス
LD_AOUT_LIBRARY_PATH	実行時の動的ライブラリパス
SHLIB_PATH	実行時の動的ライブラリパス

表- 16: コマンド・ライブラリ　パス

8.3 CVS

CVS が使う主な環境変数は表-15 に示されている。

8.4 コマンドパス

また、コマンドパス、ライブラリパスは上記のツール群の夫々の設定ファイル中で必要に応じてバイナリやライブラリがインストールされたディレクトリを追加する必要がある。

9 generating setup file

複数の shells(csh/tcsh, sh, bash, zsh,...) が利用されることを考えると、一つの設定ファイルからそれぞれの shell についての設定コマンドファイルを生成するツールが必要である。環境変数だけであれば、簡単な定義ファイルからそれらを生成するためのツールはスクリプト言語で簡単に作成出来るだろう。マクロ展開を含む環境変数の設定には注意が必要かもしれない。[6]

10 結論

参考文献

[1] "SNS Application Development Environment", Ernest Williams, http://www-group.slac.stanford.edu/cdsoft/icalepcs01/SNS-ADE2.pdf

[2] "CVS Usage in Joint Project" by K. Furukawa, http://www-linac.kek.jp/jk/cvs.html

[6] その後、古川氏とこの件について話あった。環境設定のためのスクリプトとしては機種依存性を取り扱うための条件文やファイルの存在のチェック、また不要なパスの追加を避けるなどの機能が必要。これらを汎用の設定ファイルから行うのは困難であろう。