Organisation des fichiers pour le projet T4

Luc Weber

Observatoire de Genève

1$^{\text{er}}$ juillet 1999

Table des matières

Chapitre 1

Organisation des fichiers pour le projet T4

~weber/t4/WWW/organisation_des_fichiers.tex LW/23/08/96
Les différences notées dans les marges se réfèrent à la version du 3 juillet 1996.

1.1 Buts à atteindre

Organiser les fichiers pour ce projet permet d'atteindre les buts suivants:

- Posséder une version d'exploitation stable, dite *OK*, de l'ensemble des produits.
- Posséder une version de test, dite *BETA*, permettant l'intégration des logiciels.
- Permettre au développeur de modifier sa version de développement, dite *ALPHA*, indépendamment des versions *OK* et *BETA*.
- Posséder les versions des sources correspondant aux versions des exécutables, pour permettre le debugging quelque soit la version utilisée.
- Posséder des versions figées des librairies et logiciels "extérieurs".
- Disposer de l'ensemble du softlogiciel développé à l'observatoire (unix–linux–occam–dos–...) et celui acheté (astromed–alex–nag–...) sous un seul *file system* et cela pour les versions *OK* et *BETA*.
- Faciliter la recherche des fichiers en utilisant une structure de *file system* cohérente.
- Maintenir la protection des sources sous la responsabilité du développeur (particulièrement en phase de développement).
- Utilisation et recherche des ressources simplifiée (codification des noms et des options).

- Permettre l'installation et la mise à jour sans connaissances préalable avec l'utilisation de scripts standards. Cela entre les version *ALPHA* et *BETA* et entre les version *BETA* et *OK*.

- Permettre l'utilisation simultanée de logiciels en version *OK*, *BETA* ou *ALPHA*.

Les moyens d'atteindre ces buts sont décrits dans les sections suivantes.

1.2 File system

Le *file system* est ancré sous `$T4ROOT` (actuellement `/ccd2/t4`). On y trouve les points de départ de la version *OK* et de la version *BETA*:

ok/	pour la version *OK*
beta/	pour la version *BETA*

Le répertoire de la version choisie est donné par `$T4HOME` (soit `$T4ROOT/ok`, soit `$T4ROOT/beta`)
Chacun de ces répertoires possède les répertoires standards suivant:

scripts/	fichiers scripts
bin/	fichiers binaires exécutables sous Unix
btl/	fichiers binaires exécutables pour transputer
lib/	librairies Unix
include/	fichiers *Includes* pour le C
config/	fichiers de configuration (instruments et logiciels)
prc/	procédures pour logiciels basés sur Inter
src/	fichiers sources
skeleton/	fichiers *Squelettes*

ATTENTION, il n'y a aucun développement dans ces répertoires. En effet, partant du principe qu'il y a un espace de développement (version *ALPHA*), il ne peut pas y avoir de développement en aval de cet espace (contrôle du développement).

1.2.1 Links

La structure des répertoires standards permet de retrouver ou d'adresser facilement les fichiers. Comme les fichiers qu'on y trouve sont sous la responsabilité d'un développeur, l'immense majorité des fichiers sont (et doivent être) des links vers les fichiers originaux.
Il y a cependant quelques exceptions, notament pour les librairies extérieures (ex: libnag.a) ou les fichiers de configuration des logiciels extérieurs (ex: configuration CCD d'ASTROMED), qui sont copiés (et non linkés) pour éviter des les écraser en cas de mise à jour du logiciel.

1.2.2 Répertoires dépendant de l'Operating System

Les répertoires `bin/` et `lib/` dont le contenu dépend de l'*Operating System* possèdent des sous–répertoires nommés avec le nom de l'*Operating System* et du numéro de version.
Exemples:

```
$T4HOME/bin/SunOS_5.4
$T4HOME/bin/Linux_1.2.10
```

Ce suffixe est fabriqué avec la commande Unix:

```
uname -sr | tr ' ' '_'
```

Cette manière de nommer les répertoires permet de retrouver l'exécutable du bon type ou de linker les applications avec les librairies du bon type également.

Ce suffixe est également contenu dans la variable d'environnement `$OSTYPE` (clé du système).

1.2.3 Sources

L'ensemble des logiciels développé à l'observatoire est placé sous le répertoire `src/`. Celui–ci est subdivisé en autant de répertoires qu'il y a de développeurs. Chaque répertoire porte le nom du développeur et lui appartient. Ces répertoires sont protégés en écriture.

Les fichiers sources appartiennent aux développeurs, ils sont sous leurs responsabilités et sont protégés en écriture.

Tout doit être fait pour faciliter la copie d'une application complète vers un autre répertoire. Ce dernier aspect des choses est utile dans le cas de modifications du code, lorsque par exemple, le responsable de l'application est absent. Dans ce cas là, le développeur par interim copie l'application sous un de ses répertoire et fait les modifications adéquates. Ensuite, cette nouvelle application est jointe au reste des logiciels après avoir supprimé les liens sur la version précédente.

La version stable peut suivre le même schéma à Genève mais appartiendra certainement à un utilisateur unique sur le site d'exploitation.

1.2.4 Scripts

Pour simplifier les tâches de sélection de version (*OK*, *BETA* ou *ALPHA*), de debugging (avec ou sans) et de mode de travail (simulation ou non, ...), les exécutables sous Unix ne sont jamais lancé directement, mais au travers d'un script, pointé par un link créé sous `$T4HOME/scripts`. Ainsi chaque logiciel possède un script de lancement qui reconnait l'option "`-H`" (il reconnait aussi l'option "`-h`" si l'exécutable ne la gère pas). Cette option permet au script d'indiquer de quel façon il travaille. Lorsque cette option est précisée, l'exécutable n'est pas lancé, mais apparaissent clairement tout les modes de travail possibles et les moyens de les slectionner.

Dans les répertoires source, les scripts permettant de lancer les exécutables sont nommés selon le nom de l'exécutable suivi de l'extension donnant le nom du shell (`.sh`, `.csh`, ...). Exemple:

le script lançant Inter s'appelle `inter.csh`. Par contre son link garde le nom de l'exécutable. Exemple:

```
$T4HOME/scripts/inter -> $T4HOME/src/weber/inter/inter.csh
```

Chacun des ces scripts possède une variable d'environnement associée qui permet de modifier le comportement de ce script. Le nom de cette variable est nommé comme le nom de l'exécutable. Exemple: `$INTER` pour Inter (voir plus bas sous chapître "Utilisation et Gestion des versions").

Si d'autre scripts sont liés à une application ceux–ci sont préfixé avec le nom de l'application. Par exemple `spectro_srv_init` permet de lancer les utilitaires travaillant avec `spectro_srv` (le serveur de base de donnée par exemple), bien entendu, `spectro_srv_init` est un link sur `spectro_srv_init.csh`.

1.2.5 Exécutables Unix

Si tout les exécutables sont interfacés par des scripts, le *path* des exécutables n'a pas besoin d'être nommé dans la variable d'environnement `$PATH`.
Les exécutables se trouvent dans les répertoires des développeurs et sont sufixés avec le contenu de la variable `$OSTYPE`. Exemple: `inter_SunOS_5.4`.
Les links sur ces exécutables sont créés dans le répertoire `$T4HOME/bin/$OSTYPE` (voir plus bas sous section "Maintenance globale"). Exemple:

```
$T4HOME/bin/SunOS_5.4/inter -> $T4HOME/src/weber/inter/inter_SunOS_5.4
```

1.2.6 Exécutables pour transputer

Les links sur ces exécutables sont créés dans le répertoire `$T4HOME/btl` (voir plus bas sous section "Maintenance globale").

1.2.7 Librairies Unix

Les librairies se trouvent dans les répertoires des développeurs et sont sufixés avec le contenu de la variable `$OSTYPE`. Exemple: `libgop_SunOS_5.4.a`.
Les links sur ces librairies sont créés dans le répertoire `$T4HOME/lib/$OSTYPE` (voir plus bas sous section "Maintenance globale"). Exemple:

```
$T4HOME/lib/SunOS_5.4/libgop.a ->
                              $T4HOME/src/weber/gop/libgop_SunOS_5.4.a
```

Chapitre 2

Installation et mise à jour

Une installation ou une mise à jour consiste à copier tous les fichiers relatifs à une version de logiciel, donc d'un répertoire donné, vers tout autre répertoire, notament vers le répertoire d'une version plus stable (*BETA* depuis *ALPHA* ou *OK* depuis *BETA*).
Le transfert s'appuye sur un script nommé `Copy.csh` existant dans le répertoire de l'application. Ce fichier contient:

1. les ordres de création des répertoires nécessaires à l'application
2. la liste des fichiers nécessaires à l'application
3. toute autres commandes nécessaires à la copie de l'application (et non pour sa compilation)

On l'utilise de la manière suivante:

```
% cd <destination_répertoire>
% cp <source_répertoire>/Copy.csh .
% ./Copy.csh
```

Durant son exécution, `Copy.csh` crée les répertoires s'ils n'existent pas, puis fabrique un fichier de type *Makefile* contenant les ordres de copie de tout les fichiers donnés dans une liste interne au script. Ce *Makefile* est ensuite exécuté, générant uniquement les copies nécessaires lorsque le fichier d'origine est plus récent que le fichier local, et finalement exécute les commandes associées à l'application. Ainsi il est possible de lancer plusieurs fois de suite ce type de script, les copies se faisant uniquement une fois.
Le fichier `Copy.csh` est naturellement différent dans chaque répertoire. Chaque développeur utilise un fichier *Squelette* donné dans `$T4HOME/skeleton/Copy.csh.skeleton` pour chacune de ses applications.

Remarque: tout les fichiers `Copy.csh` contiennent le *path* de l'application. Par exemple le script `~weber/src/gop/Copy.csh` contient la ligne:

```
set ORG=/home/ccd/weber/src/gop
```

Ce *path* est utilisé pour faire les copies, il est nécessaire car on lance toujours la copie depuis le répertoire de destination. Le fichier `Copy.csh` est auto–édité en fin d'exécution pour refléter de nouveau répertoire source. Par exemple, apres avoir lancer la copie depuis `$T4HOME/src/weber/gop`, le fichier `Copy.csh` contient maintenant:

```
set ORG=/ccd2/t4/beta/src/weber/src/gop
```

Ansi, ce script est réutilisable pour effectuer des copies de copie et ainsi de suite.

Installation et Mise à jour d'applications basées sur Inter

Le cas d'installation et de mise à jour d'applications basées sur Inter est différente car le fichier `Copy.csh` existe déjà et il sert à copier le noyau d'Inter depuis un répertoire de référence. Ce fichier est sous la responsabilité du responsable du noyau d'Inter (Weber). Il doit donc exister un fichier équivalent à `Copy.csh` servant uniquement à copier les fichiers hors du noyau d'Inter nécessaire à l'application. Ce fichier se nomme `Copy.<application>.csh` et travaille de la même manière que `Copy.csh` (avec auto–édition en fin de travail).
Chaque développeur utilise un fichier *Squelette* donné dans `$T4HOME/skeleton/Copy.application.csh.skeleton` pour chacune de ses applications basées sur Inter.
Avec les scripts `Copy.csh` et `Copy.<application>.csh` on peut copier l'application complète vers un autre répertoire indépendament de niveau de développement du noyau d'Inter, en effet on recopie le noyau tel qu'on l'a obtenu par une copie précédente.

Chapitre 3

Compilation

Toutes les compilations sous Unix sont basées sur l'usage de make. Chaque application contient dans son répertoire de base un fichier nommé "`Makefile`" ou "`makefile`". Il suffit donc de taper `make` dans un répertoire pour que l'application soit opérationelle.

Pour permettre la création de librairies ou d'exécutables pour différents *Operating Systems* ou différentes versions d'un *Operating System* à partir d'un *Makefile* unique et les maintenir dans un même répertoire, la stratégie suivante est adoptée:

- Le fichier *Makefile* inclus un fichier de type *Rulefile* nommé selon l'*Operating Sytem* sur lequel on travaille (syntaxe: `Rules_$OSTYPE.mke`). Exemple: `Rules_SunOS_5.4.mke`. Ce fichier contient les options des compilateurs et du *linker*, c'est à dire principalement les *paths* pour les fichiers *Includes* et pour les librairies ainsi que les noms des librairies.
- les noms des exécutables et des librairies sont suffixées par `$OSTYPE`. Par exemple: `inter_SunOS_5.4`, `libgop_Linux_4.1.10.a`.
- Les cibles des *Makefiles* sont les noms des exécutables et des librairies donnés juste ci–dessus.

Pour permettre la compilation globale de l'ensemble des librairies et des exécutables de tout les développeurs, il est absolument obligatoire de construire d'abord toutes les librairies puis tous les exécutables. Ceci est possible si tous les *Makefiles* réagissent aux commandes suivantes:

```
% make all_lib
% make all_exe
```

Pour permettre cela, les *Makefiles* doivent contenir les lignes suivantes:

```
all: all_lib all_exe

all_lib: <liste des librairies construites pour cette application>
all_exe: <liste des exécutables construits pour cette application>
```

Une des deux listes peut naturellement être vide.

Chaque *Makefile* et *Rulefile* doit être écrit de telle sorte que l'on puisse les utiliser sans modifications sous n'importe quelle version (*ALPHA*, *BETA* ou *OK*) et si possible sous n'importe quel *Operating Sytem*.

Pour réaliser un *Rulefile*, le *Squelette* suivant peut être utilisé (voir `$T4HOME/skeleton/Rules.skeleton`):

```
#
# definitions specifiques a l'operating system
#
T4INC   = $(T4HOME)/include
T4LIB   = $(T4HOME)/lib/$(OSTYPE)

INCLUDE_PATH =  -I .                          \
                -I $(INC_DIR)                 \
                -I $(T4INC)                   \
                -I $(OPENWINHOME)/include     \
                -I /usr/local/include

LIBRARY_PATH =  -L .                          \
                -L $(LIB_DIR)                 \
                -L $(T4LIB)                   \
                -L $(OPENWINHOME)/lib         \
                -L /usr/local/lib

CFLAGS  += -g -DSYSV $(INCLUDE_PATH)

LDFLAGS += $(LIBRARY_PATH)

LDLIBS  = -lgop -ltpudummy -lxview -lolgx -lX -lsocket -lnsl

RANLIB  = echo
```

Avec ces déclarations, pour les fichiers *Includes* et pour les librairies, on accède dans l'odre:

1. les fichiers faisant partie de l'application
2. les fichiers nécessaires à l'application créés par le développeur
3. les fichiers nécessaires à l'application créés par les autres développeurs
4. les fichiers système ou les fichiers de logiciel installés

Le développeur désirant utiliser à un moment spécifique une version précise d'une librairie (*ALPHA*, *BETA* ou *OK*) doit temporairement faire un link dans son répertoire `lib/$OSTYPE` sur la librairie désirée.

Ce système marche dans tout les cas à condition que le développeur initialise dans son fichier `~/.login.local` les variables T4ROOT, T4HOME, LIB_DIR et INC_DIR. Par exemple si on veut utiliser la version stable des logiciels des autres développeurs:

```
setenv T4ROOT    /ccd2/t4
setenv T4HOME    /ccd2/t4/ok
setenv LIB_DIR   $HOME/lib/$OSTYPE
setenv INC_DIR   $HOME/src/incl
```

ou s'il l'ont veut utiliser la version beta des logiciels des autres développeurs:

```
setenv T4ROOT    /ccd2/t4
setenv T4HOME    /ccd2/t4/beta
setenv LIB_DIR   $HOME/lib/$OSTYPE
setenv INC_DIR   $HOME/src/incl
```

Il faut remarquer que si le développeur gère plusieurs applications, il est alors nécessaire qu'il possède une structure de répertoires analogue à celle que l'on retrouve pour T4, en tout cas pour les librairies et les *Includes*. Dans ce cas il doit utiliser des links fabriqués tout naturellement avec `Dolink.csh` (voir plus bas dans "Maintenance globale").

Lors d'une mise à jour des versions beta ou ok, **en utilisant les environnement adéquats (voir `UGTOOL`)**, ces variables sont initialisées avec les valeurs suivantes: (exemple pour la version *BETA*)

```
setenv T4ROOT    /ccd2/t4
setenv T4HOME    /ccd2/t4/beta
setenv LIB_DIR   $T4HOME/lib/$OSTYPE
setenv INC_DIR   $T4HOME/include
```

Pour l'écriture du *Makefile*, il faut donner les dépendances avec leur *path* complet.

L'exemple suivant montre un *Makefile* complet qui permet de créer un exécutable nommé `server` et qui utilise trois librairies, la première (`libserver.c`) fait partie de l'application et inclus les fichiers `server.h`, `gop.h` et `tpu.h`, la seconde (`libgop`) est aussi développée par le même développeur et la troisième (`libtpu`) fournie par un autre développeur (`gop.h` est lié a `libgop` et `tpu.h` est lié à `libtpu`).

Exemple (voir `$T4HOME/skeleton/Makefile.skeleton`):

```
# Makefile pour server
# --------------------
#
# OSTYPE doit contenir:  <operating_system>_<version>
#              exemple:  SunOS_4.1.3, SUNOS_5.3, Linux_1.1.18
#
# definitions specifiques a l'operating system
#
include Rules_${OSTYPE}.mke
#
# targets specifiques a l'operating system
#
SERVER    = server_${OSTYPE}
LIBSERVER = libserver_${OSTYPE}.a

.KEEP_STATE:
.PRECIOUS: $(LIBSERVER)

all: all_lib all_exe

all_lib: $(LIBSERVER)
all_exe: $(SERVER)

$(SERVER): server.c              \
           $(LIBSERVER)          \
           $(LIB_DIR)/libgop.a   \
           $(INC_DIR)/gop.h      \
           $(T4LIB)/libtpu.a     \
           $(T4INC)/tpu.h
        cc $(CFLAGS) $(LDFLAGS) server.c -o $(SERVER) \
                               $(LIBSERVER) $(LDLIBS)


$(LIBSERVER):  $(LIBSERVER)(libtpu.o) \
               server.h
        $(RANLIB) $(LIBSERVER)
```

3.1 Maintenance globale

Chaque développeur doit maintenir l'ensemble des applications pour lesquelles il est responsable. C'est à dire qu'il doit pouvoir

1. installer et mettre à jour les versions *BETA* et *OK*.
2. compiler les applications dans les versions *ALPHA*, *BETA* et *OK*.
3. effectuer les links des fichiers dans les versions *ALPHA*, *BETA* et *OK*.

Grâce aux scripts de copie et aux *Makefiles*, ces opérations sont très simplifiée. Toutefois pour faciliter encore plus la tâche, quelques scripts, à nouveau sous la responsabilité du développeur, sont utilisés pour les diverses étapes de la maintenance. Ce sont:

Dolink.csh	effectue l'ensemble des links
Install.csh	effectue l'ensemble des copies
Compile_lib.csh	compilation des applications fournissant une librairie
Compile_exe.csh	compilation des applications

Ces scripts peuvent être plus ou moins complexes. Chaque développeur en possède une version personalisée dans son répertoire de base. Chaque développeur doit se baser sur les fichiers *Squelettes*, donné dans le répertoire `$T4HOME/skeleton`, pour gérer la maintenance de l'ensemble de ses applications.

Remarque: dans le script `Install.csh` le répertoire d'origine (version *ALPHA*) et le répertoire d'intégration (version *BETA*) sont codés. Si ce script est lancé depuis un répertoire dont le *path* contient la chaine de caractère "/ok/", le script considère que l'on installe la version *OK* et il effectue la copie de la version *BETA*, autrement ce script copie la version *ALPHA*.

Chapitre 4

Utilisation et Gestion des versions

Chaque utilisateur (développeur ou non) qui veut utiliser le logiciel t4 dans sa globalité doit travailler dans un environnement ad hoc. Cet environnement définit le *path* des scripts (et non celui des exécutables), le répertoire de base de la version choisie (*OK* ou *BETA*) ainsi que diverses variables d'environnement dont le développeur a besoin lors des mises–à–jour.

Deux environnements pour les versions *BETA* et *OK* sont disponibles sous `ugtool` ou par l'intermediaire de scripts `t4.csh` et `t4_beta.csh` (Voir sous `$T4HOME/scripts`). On les utilise ainsi:

```
% source $T4HOME/scripts/t4.csh
% source $T4HOME/scripts/t4_beta.csh
% ugtool t4
% ugtool t4_beta
```

La version choisie par défaut est celle définie par l'environnement, c'est à dire soit la version *OK*, soit la version *BETA*.
Les scripts de lancement d'application permettent d'utiliser les logiciels des autres versions (*BETA* ou *ALPHA* si on utilise la version *OK*; ou *OK* ou *ALPHA* si on utilise la version *BETA*). Ce choix s'effectue en initialisant les variables d'environnement associées au script.
Par exemple, si en travaillant sur la version *OK* on désire utiliser la version *ALPHA* de Inter, on tape simplement:

```
% setenv INTER alpha
% inter
```

4.1 Debugging

Le debugging d'un exécutable sous Unix est lancé par le script associé à l'exécutable. C'est le contenu de la variable d'environnement associées au script qui donne l'ordre de lancer le debugger.
Par exemple, si en travaillant sur la version *OK* on désire debugger la version *ALPHA* de Inter, on tape simplement:

```
% setenv INTER alpha+debug
% inter
```

4.2 Définitions globales

Avec l'utilisation des scripts de lancement, comme les versions des exécutables prisent par défaut sont soit *OK* soit *BETA*, le développeur ou tout autre personne, qui désire utiliser un exécutable d'une version particulière , doit définir toutes les variables d'environnement correspondantes à la version désirée.
Pour faciliter la tâche, les valeurs par défaut de ces variables peuvent être écrites dans un fichier sous le format:

```
<VARIABLE_NAME> <default_value>
```

Exemple:

```
INTER        alpha
SPECTRO_SRV  simul+alpha
CCD          beta
```

Pour que ce fichier soit pris en compte, il faut donner son *path* dans la variable d'environnement `$INITENV`.

Lors de l'utilisation des environnements sous `UGTOOL`, la variable `$INITENV` est annulée, ainsi le développeur qui utilise un tel environnement, travaille vraiment avec la version désirée (*OK* ou *BETA*).

Annexe A

Variables d'environnements

Lorsqu'on travaille sous les versions *OK* ou *BETA* les variables d'environnement définies sont les suivantes:

$INC_DIR	== $T4HOME/include
$LIB_DIR	== $T4HOME/lib/$OSTYPE
$EXE_DIR	== $T4HOME/bin/$OSTYPE
$SCR_DIR	== $T4HOME/scripts
$BTL_DIR	== $T4HOME/btl
$CON_DIR	== $T4HOME/config
$PRC_DIR	== $T4HOME/prc
$SKE_DIR	== $T4HOME/skeleton

Les noms de ces variables n'incluent pas le mot "T4" car la plupart des logiciels sont indépendant du projet T4.

Ces variables sont utilisées dans certains scripts d'installations (ex: `Dolink.csh`) et dans les fichiers de type *Makefile* et *Rulefile*.
L'utilisation de ces variables dans les fichiers Makefile et *Rulefile* permettent de compiler les programmes C avec les fichiers *Includes* de la version correspondante et de linker les *objets* avec les librairies de la version correspondante d'une manière totalement transparente.

A.1 Gestion de la version de développement *ALPHA*

La définition des variables d'environnement données ci–dessus doit être faite par le développeur pour la version *ALPHA*. Ces définitions se font dans le fichier `~/.login.local`.
Pour simplifier le gestion de ses propres développements, il est fortement conseillé que le développeur crée une structure de répertoire en partie équivalente à celle de t4. Elle lui permet d'utiliser les scripts d'installation (`Dolink.csh`, `Install.csh`, `Compile_lib.csh` et `Compile_exe.csh`) d'une manière complètement transparente.
Dans ce cas la structure minimum est:

scripts/	fichiers scripts
bin/$OSTYPE	fichier binaires exécutables
lib/$OSTYPE	librairies
include/	fichier *Includes* pour le C

Tout comme pour la structure t4, ces répertoires ne contiennent que les links fabriqués par `Dolink.csh`.

A.1.1 Exemple de fichier `~/.login.local`

Pour un développeur travaillant sur sa version *ALPHA*, le fichier `~/.login.local` a cette allure–ci:

```
setenv T4ROOT /ccd2/t4
setenv T4HOME $T4ROOT/ok

setenv OSTYPE    `uname -sr | tr ' ' '_'`
setenv PATH       .:"$HOME":"$HOME/scripts":"$HOME/bin/$OSTYPE":\
                  "$T4HOME/scripts":$PATH
setenv DEBUGGER  debugger

setenv INC_DIR   "$HOME/src/incl"
setenv LIB_DIR   "$HOME/lib/$OSTYPE"
setenv EXE_DIR   "$HOME/bin/$OSTYPE"
setenv SCR_DIR   "$HOME/scripts"
setenv BTL_DIR   "$HOME/btl"
setenv CON_DIR   "$HOME/config"
setenv PRC_DIR   "$HOME/prc"
setenv SKE_DIR   "$HOME/skeleton"

setenv INITENV   "$HOME/src/Initenv"
```

Ce fichier est normalement le seul endroit ou le développeur doit faire des modifications en cas de restructuration.

Annexe B

Exemple de Structure

```
/ccd2/t4/ok       ($T4HOME)

        lib/
                SunOS_5.4/
                        libtpu.a        -> $T4HOME/src/maire/t120_srv/libtpu_SunOS_5.4.a
                        libgop.a        -> $T4HOME/src/weber/gop/libgop_SunOS_5.4.a
                Linux_1.2.10/
                        libgop.a        -> $T4HOME/src/weber/gop/libgop_Linux_1.2.10.a
        bin/
                SunOS_5.4/
                        t120            -> $T4HOME/src/maire/t120/t120_SunOS_5.4
                        t120_srv        -> $T4HOME/src/maire/t120/t120_SunOS_5.4
                        inter           -> $T4HOME/src/weber/inter/inter_SunOS_5.4
                Linux_1.2.10/
        btl/
        include/
                gop.h -> $T4HOME/src/weber/libgop/gop.h
        scripts/
                t120                    -> $T4HOME/src/maire/t120/t120.csh
                run_t120_srv.csh        -> $T4HOME/src/maire/t120_srv/run_t120_srv.csh
                init_t120_srv.csh       -> $T4HOME/src/maire/t120_srv/init_t120_srv.csh
                inter                   -> $T4HOME/src/weber/inter/inter.csh
        config/
                ccd/
                        Kodak1400/
                                normal.cfg      -> $T4HOME/...
                        p88332/
                                normal.cfg      -> $T4HOME/...
                t120/
                        azi.sdb         -> $T4HOME/...
                        ele.sdb         -> $T4HOME/...
                spectro/
                        spectro.sdb     -> $T4HOME/src/weber/spectro_srv/spectro.sdb
        prc/
```

```
        inter/
        common/
        ccd/
                torture.prc     -> $T4HOME/src/simond/ccd/prc/torture.prc
        spectro/
                torture.prc     -> $T4HOME/src/weber/spectro/prc/torture.prc
        t120/
                torture.prc     -> $T4HOME/src/maire/t120/prc/torture.prc
        tacos/
src/

        weber/
                Install.csh
                Compile_lib.csh
                Compile_exe.csh
                Dolink.csh
                ccd/
                        Makefile
                        Rules_SunOS_5.4.mke
                        ccd_SunOS_5.4
                        <sources>
                        tex/
                                <doc latex>
                                ccd_reference/
                                        <doc HTML>
                chrono/
                gop/
                        Makefile
                        Rules_SunOS_5.4.mke
                        Rules_Linux_1.2.10.mke
                        libgop_SunOS_5.4.a
                        libgop_Linux_1.2.10.a
                        <sources>
                        tex/
                                <doc latex>
                                gop_reference/
                                        <doc HTML>
                inter/
                ipc/
                ipcsrv/
                ipcstat/
                libaff/
                libgsc/
                libipc/
                logbook/
                sdb/
                spectro/
```

```
                subgsc/
                xaff/
                xdbox/
                xgsc/
        maire/
                Install.csh
                Compile_exe.csh
                Dolink.csh
                t120/
                t120_srv/

        simond/
                db-1.79/
                include/
                libamc/
                libsrv/
                sccd/
        blecha/
        russinie/
        t4/
        queloz/
        jolissai/
                wfa/
alex/
        <distribution alex (tpu)>
astromed/
        <distribution astromed (ccd)>
coralie/
        ohp/
                <distribution coralie originale>
```