Inter
&
Autres Applications

Guide de l'administrateur

Luc Weber
Observatoire de Genève

24 octobre 2012

Table des matières

Chapitre 1

Gestion sur site

Inter est un utilitaire d'usage général composé d'un interpréteur et de *commandes internes* formant un noyau. Ce noyau est utilisé pour former des applications en lui ajoutant des commandes spécifiques dites *commandes locales.*
Actuellement il est utilisé par Inter–Moan qui permet le contrôle du télescope Genevois de La Silla au Chili ainsi que le traitement d'images CCD et, Inter–Tacos, le logiciel de Traitement, d'Analyse et de Corrélation de Spectre sur le spectromètre ELODIE à St–Michel.
Ainsi lorsque l'on parle d'Inter dans ce manuel, on parle du noyau, sinon on précise le nom complet de l'application à laquelle il est lié ou on utilise le terme *Application.*

Une *Application* travaille conjointement avec d'autres logiciels (en majorité des serveurs) et utilise des libraires spécifiques. L'ensemble de ces logiciels et librairies est regroupé dans des répertoires que l'on nommera d'une manière générale <`application_directory`>. Ces répertoires sont placés sous un répertoire commun que l'on appellera <`base_directory`>. La place nécessaire pour l'ensemble des logiciels et d'environ 30[MB].

Les <`application_directory`> sont les suivantes :

inter	application d'Inter
libaff	librairie d'accès au server xaff
libgsc	librairie d'accès au server xgsc
libipc	librairie de gestion des sémaphores
gop	librairie du protocole de communication GOP
xaff	afficheur sous OpenWindow
xgsc	générateur de carte stellaire (avec GSC) sous OpenWindow
subgsc	librairie pour l'accès à la BDD de GSC
incl	fichiers de type "inclu"
ipcsrv	server de remote sémaphores
ipcstat	moniteur de sémaphores
xdbox	générateur de panneaux interactifs
logbook	utilitaire de log
notice	afficheur de notice
panel	afficheur de message

1.1 Environnement de développement

L'ensemble des fichiers regroupés sous le répertoire de base et conçu pour être géré par un seul utilisateur et donc un seul responsable que l'on appellera ici le développeur. Ces répertoires contiennent les versions de développement de chaque logiciel. Le terme logiciel inclus ici tout les exécutables, libraires et scripts.

Les utilisateurs n'ont pas accès au logiciels se trouvant dans le répertoires de développement. Ils accèdent les logiciels testés qui sont placés dans des répertoires communs. Le développeur possède certains outils (scripts et makefile) pour la mise en commun aisée des logiciels (exécutables, librairies et scripts). Ces outils permettent la mise en commun des logiciel dans un réseau comportant un ou plusieurs serveurs d'applications sous différents systèmes d'exploitations (OS).

1.1.1 Relation avec le système d'exploitation

L'environnement logiciel dans lequel se place le développeur est défini par le ou les OSs sous lesquels fonctionnent les logiciels et par le système de fichier qui définit les répertoires où se placent les logiciels, qu'ils soient de développement ou qu'ils soient en commun.

Par exemple : un réseau local peut avoir simultanément des machines travaillant sous SunOS 4.1.3, sous SunOS 5.4 (Solaris) ou sous Linux 1.1. Les répertoires commun dépendent de cet OS. Pour placer les exécutables, on a par exemple `/import/bin` sous SunOS 4.1.3, `/usr/local/bin` sous SunOs 5.4 et `/usr/local/bin` sous Linux 1.1.
Cette situation oblige le développeur à fabriquer des exécutables et des librairies dépendant des différents OSs sous lequel il travaille.

Pour éviter de duplifier les sources des programmes pour chaque OS, la convention suivante à été appliquée pour nommer les exécutables, les objets et les librairies et ainsi les faire cohabiter dans un même répertoire :

La commande UNIX `uname` permet de connaître le nom de l'OS :

```
> uname -sr
SunOS 5.5.1
```

après avoir supprimé le troisième digit (on ne s'interresse qu'au No de release et au No de sous release) et remplacé les espaces du texte retourné par des "_", on obtient un mot qui devient le suffixe *officiel* qui définit l'OS :

```
> uname -sr | cut -d. -f1-2 | tr ' ' '_'
SunOS_5.5
```

Toutefois, pour éviter de posséder des exécutable pour différents OS qui sont pleinement compatibles (ex : SunOS_5.4 et SunOS_5.5), on force la définition de l'OS à un OS prédéfini. Par exemple :

```
setenv OPSYS ‘/bin/uname -sr | cut -d. -f1-2 | tr ’ ’ ’_’‘
if ( $OPSYS == SunOS_5.5 ) then
      setenv OPSYS "SunOS_5.4"
endif
```

Chaque nom est suffixé (le séparateur est le "_") avec ce mot. Par exemple, dans le répertoire d'inter, on trouve les exécutables `inter_SunOS_4.1` et `inter_SunOS_5.4`. Pour la librarie libinter, on a `libinter_SunOS_4.1.a` et `libinter_SunOS_5.4.a`.

Accès aux exécutables

Pour faciliter l'appel aux exécutable, c'est à dire les appeller toujours de la même manière quelle que soit l'OS, le développeur installe sous son *home–directory* les répertoires (par exemple) : `~/bin/SunOS_4.1` et `~/bin/SunOS_5.4`. Dans ces répertoires sont crées des links vers les exécutable.
Par exemple : `~/bin/SunOS_5.4/inter` pointe sur `~/src/inter/inter_SunOS_5.4`. Pour que ces répertoires soient dans le path des exécutables du développeur, les lignes suivantes sont à rajouter dans le fichier de login du développeur (".login.local" par exemple) :

```
setenv OPSYS ‘uname -sr | tr ’ ’ ’_’‘
set path=(. $home/scripts $home/bin/$OPSYS $path)
```

La veriable OPSYS contient le nom de l'OS et sa version.
On remarque que le répertoire `$home/scripts` permet d'accéder les scripts, qui eux ne dépendent pas de l'OS (du moins pour l'appel).

Accès aux libraries

Les libraries sont aussi dépendantes de l'OS. Pour permettre un accès transparent au libraries, on fabrique un *link* pour chaque libraries depuis des répertoires spécifiques à l'OS. Par exemple : `~/lib/SunOS_4.1` et `~/lib/SunOS_5.4`
Pour que ces répertoires soient reconnus au niveau des Makefile, la variable d'envirronement `LIB_DIR` doit être définie dans fichier de login du développeur (".login.local" par exemple) :

```
setenv LIB_DIR "/home/ccd/weber/lib/$OPSYS"
```

Accès aux fichiers includes

Pour que le développeur puisse accéder les fichiers de type inclus de la version de dévellopement, on crée des liens sur ceux–ci dans un répertoire nommé par la variable d'envirronnement `INC_DIR` dans fichier de login du développeur (".login.local" par exemple) :

```
setenv INC_DIR "/home/ccd/weber/solaris/incl"
```

1.1.2 Mise à jour dans un répertoire

Chaque répertoire possède un fichier makefile permettant de contruire le ou les exécutables et/ou librairies du répertoire courant. Ce makefile inclus un fichier de definition relatif à l'OS. Ce fichier est fabriqué avec le nom de l'OS, de sa version et a comme extension "mke". Par exemple, il s'appelle : "Rules_SunOS_4.1.mke" pour SunOS 4.1 ou "Rules_SunOS_5.4.mke" pour SunOS 5.4. En cas de modification locales, ce sont les fichiers `Rules_*` qu'il faut modifier et non le `Makefile`.
L'appel à ce makefile doit se faire dans le répertoire local, par exemple :

```
> cd <application_directory>
> make
```

Chapitre 2

Installation d'une application

Une *Application* fonctionne uniquement si la variable d'environnement INTERHOME existe et contient le répertoire de base de celle-ci. L'assignation se fait de la manière suivante. Exemple :

```
> setenv INTERHOME /src/inter
```

Comme vu précédement *Application* et composée de 2 parties.
- Le noyau avec ses commandes **internes** permettant les opérations courantes : accès aux blocs de données, aux variables, aux fichiers, etc... et les commandes de contrôles : do–enddo, if-endif, etc...
- Les commandes **locales** permettant toutes les commandes spécifiques à une application : opérations sur images, de commande de télescope, de réduction de spectres, etc...

La distinction entre ces deux types de commandes n'est pas franche, mais le noyau d'inter et les sources des commandes internes sont localisées dans le répertoire `$INTERHOME`, alors que toutes les commandes locales ont leurs sources dans le répertoire `$INTERHOME/for`.
Une *Application* possède les répertoires suivants :

`$INTERHOME`	pour les sources du noyau et des commandes internes
`$INTERHOME/for`	pour les sources des commandes locales
`$INTERHOME/hlp`	pour les fichier d'aide
`$INTERHOME/tex`	pour les sources des manuels en LaTex
`$INTERHOME/exe`	pour les exécutables lancés par certaines commandes d'inter
`$INTERHOME/tmp`	pour les fichiers intermédiaires

2.1 Installation sans compilation

Cette section décrit la marche a suivre pour l'installation d'une *Application* livrée "clé en main". C'est à dire que l'*Application* et ses serveurs sont livré sous la forme d'exécutable avec uniquement les fichiers de help et les scripts.

Puisque l'exécutable est fourni, aucune compilation n'est nécessaire. Par contre, il faut tester son bon fonctionnement. Le seul problème à ce stade est de vérifier si l'*Application* trouve toutes ses librairies partagées. Pour tester cela, on définit `INTERHOME` puis on lance l'*Application* :

```
> cd <application>
> setenv INTERHOME ‘pwd‘
> <application>
```

Si le programme répond, c'est qu'il est utilisable immédiatement. S'il indique qu'il ne trouve pas des librairies partagées il faut utiliser la commande :

```
> ldd <application>
```

qui montre la totalité des librairies partagées utilisées par un programme. Il faut alors les rechercher (voir avec le gourou local) et enfin compléter la variable d'environnement `LD_LIBRARY_PATH` avec les nouveau path permettant d'accéder ces librairies.

```
> setenv LD_LIBRARY_PATH $LD_LIBRARY_PATH":<path1>[:<path2>...]"
```

2.2 Réinstallation

La réinstallation se déroulle lorsque l'on désire refabriquer une *Application* à partir des sources. Il faut tuer tout les objets et libraries dans `$INTERHOME` puis recompiler l'utilitaire de génération **inicode**, le lancer et compiler l'*Application.* C'est à dire :

```
> cd $INTERHOME
> rm *.o *.a
> make
```

A ce stade, l'*Application* est prête.