Titre
20150407

F. Meyer
LTFB/UTINAM/OSU-THETA/UFC
fm@ltfb.fr

1er novembre 2018

1 Structuration

Une approche "pilotée par le projet" est peut-être la seule solution capable de couvrir tous les besoins (en gros, une approche où on commence par décrire le projet en amont, exprimé par une structure (un arbre par exemple, voir ci-dessous) qui hiérarchise l'ensemble des opérations à effectuer pour mener le projet à bien : si on prend le projet de Colmic comme cobaye pour tester l'efficacité de nos approches, on se dit qu'est-ce que Colmic doit fournir comme informations indispensables (qui sont ses décisions de réalisation de son projet) ? Et on essaie de distinguer les élément qui relèvent de ses décisions d'observateur des éléments qui sont des règles systématiques, répétables qui n'ont besoin d'aucun input de l'observateur.

Venons-en à notre arbre. J'écris ça avec en arrière-pensée une logique de Makefile : le produit final dépend d'un certain nombre d'étapes intermédiaires, et chacune de ces étapes est un sous-projet qui dépend également d'étapes intermédiaires, la récursion s'arrêtant lorsqu'une étape se traduit par la création (NEWTASK) d'une tâche de production (acquisition ou traitement) d'une image ou d'une séquence d'images.

Colmic veut faire une image LRGB (pour le moment on simplifie, on verra comment on peut ajouter des temps de pose différents, d'autres FREQ par la suite) : la production de l'image LRGB nécessite l'existence des 4 composantes LImage RImage GImage BImage : Comme dans un makefile, on liste d'abord la cible puis, après les " :", les objets dont dépend la cible, puis à la ligne suivante, les actions à mener pour fabriquer la cible à partir des objets s'ils existent (s'ils n'existent pas, on doit créer une règle qui explique comment les fabriquer).

Donc on commence par l'objet final, on écrit de quoi on a besoin pour la fabriquer, et on continue récursivement (pour le moment on cherche pas à être intelligent, on fait juste de la mécanique récursive pour voir et estimer ce que ça donne, on essaiera d'être intelligent après (comme de systématiser Image Rimage GImage sous la forme de FREQImage par exemple, ou FREQ deviendrait une variable pouvant être tout ce qu'on veut (LRGB Ha, OIII, ...) ; pour le moment on ne se préoccupe que de la structure, pas de l'implémentation effective.

Je résume la démarche : on crée un projet sous forme de Makefile, on mouline le makefile qui génère la liste des tâches à accomplir pour réaliser le projet. Cette liste contiendra les acquisitions et les traitements, charge au logiciel d'acquisition de trier ce qui le concerne. On pourrait aussi créer deux listes distinctes, peu importe à ce stade.

Première étape, la création du Makefile.

Première entrée, entrée principale, celle où l'observateur dit ce qu'il veut faire, et a priori la seule ou son input est nécessaire, toutes les autres entrées sont génériques, créées automatiquement au moment de la définition du projet (ou copiées d'un Makefile générique par exemple) :

```
LRVBImage: LImage RImage GImage BImage
   NEWTASK PROCESS compositer LRGB avec input LImage RImage GImage BImage
```

Les entrées suivantes définissent comment sont construites chacune des composantes du projet (je détaille pas, mais ci-dessous, les lignes FREQ doivent être déclinées pour chaque valeur souhaitée de FREQ (L, R, G, B, ...) ; voir au moment de l'implémentation comment on gère ça dans un cadre "Makefile"-like.

```
FREQImage: FREQ_Light_seq MasterBias MasterDark MasterFREQFlat
   NEWTASK PROCESS Script de preprocess de la sequence FREQ_Light_seq

FREQLight_seq:
   NEWTASK ACQ FREQ=Light TASKID=FREQLight_seq

MasterBias: Bias_seq
   NEWTASK PROCESS Stack Bias_seq

Bias_seq:
   NEWTASK ACQ sequence Bias_seq

MasterDark: Dark_seq
   NEWTASK PROCESS Stack sequence dark

Dark_seq:
   NEWTASK ACQ sequence Dark

MasterFREQFlat: FREQFlat_seq MasterBias
   NEWTASK PROCESS FREQFlat_seq  MasterBias

FREQFlat_seq:
   NEWTASK ACQ SEQ FREQ=FREQ TASKID=FREQFlat_seq
```

On essaie de dérouler ça comme un Makefile Si je tape make LRVBImage : on cherche LImage, n'existe pas, mais il y a une règle pour la fabriquer, la ligne FREQImage : j'ai besoin de

- FREQ_Light_seq
- MasterBias
- MasterDark
- MasterFREQFlat

FREQ_Light_seq n'existe pas, mais j'ai une règle pour la fabriquer qui me dit comment je la fabrique :

```
ACQ sequence FREQLight
```

je stocke ça dans une liste des actions à effectuer, elle sera sélectionnable depuis l'acquisition par l'observateur qui pilote le process.

Je passe à MasterBias, qui n'existe pas, la règle me dit que j'ai besoin de `Bias_seq` ; `Bias_seq` n'existe pas, la règle pour la construire est

```
ACQ sequence Bias
```

j'ajoute ça à la liste des tâches et je marque `Bias_seq` comme "planifiée" (c'est à dire "terminée" du point de vue du Makefile, dont l'objectif est planifier les tâches) ; je continue à dérouler, une fois que `Bias_seq` existe (ou plus exactement est "planifiée", c'est à dire que les tâches pour la construire ont été créées), pour construire `MasterBias` je dois faire `"PROCESS Stack sequence bias"` ; j'ajoute ça à liste des tâches, je marque `MasterBias` comme "planifié" (terminé au sens du Makefile) et je continue : même process pour `MasterDark`, qui aboutit à la création de 2 tâches `ACQ sequence Dark` et `PROCESS Stack sequence dark`

On passe à MasterLFLAT (FREQ=L) qui a besoin de `FREQFlat_seq` et `MasterBias` pour `FREQFlat_seq`, la règle fournit une action : `ACQ sequence FREQFlat` (ou `ACQ sequence LFLAT` une fois instanciée) qu'on ajoute à liste. `MasterBias` est déja marquée "planifiée", rien à faire de plus.

Le process se répète à l'identique pour chaque FREQ R, G, B.

Si je n'ai rien oublié, à la fin j'ai une liste de tâches ACQ et PROCESS ; les tâches ACQ sont indépendantes les unes des autres, donc le soft d'acquisition peut en fournir une liste déroulante à l'observateur pour qu'il choisisse celle qu'il veut effectuer.

Les tâches PROCESS constituent pour l'essentiel le script de traitement : a priori le process de Makefile les crée, par construction, dans un ordre cohérent (à vérifier ; on peut vérifier rapidement que l'action "PROCESS compositer LRGB avec input LImage RImage GImage BImage" est bien la dernière à être insérée dans la liste des tâches).

Et donc la commande finale c'est : siril -s ./lenomduscriptquonacree

Voilà pour la structure, à grands traits. Ça doit pouvoir être fait avec GNU make, et un script NEWTASK qui écrit comme il faut ce qui doit l'être, en fonction des arguments.

2 Implémentation

Au-delà de l'implémentation sous forme de Makefile, les entêtes FITS fournissent une base prometteuse pour permettre de tracer tout ce qui doit l'être dans le processus d'acquisition/traitement qui doit mener à l'image finale.

On a besoin d'être capable d'identifier la nature de toutes les images.

2.1 Entête FITS

2.1.1 KEYWORDS usuels

— IMAGETYP
— EXPTIME
— FREQ

2.1.2 KEYWORDS spécifiques

1. TASKID : integer TASKID permet au logiciel d'acquisition de reprendre une séquence existante pour y ajouter des éléments plutôt que de créer une nouvelle séquence ; TASKID est créée par le makefile.
2. SEQNUM : integer 0 = single image ;

Contrainte, le logiciel d'acquisition doit être en mesure de vérifier si une séquence de même nature existe déjà dans la même session : PROJECT IMAGETYP SESSION FREQ EXPTIME

2.2 Traitement

Un peu plus dans le détail, les entêtes FITS sont utilisées pour garder la trace de tout ça dans les fichiers au fur et à mesure que le processus progresse.