M1 ALMA/ATAL
X7II030

Université de Nantes
2015/2016

# *Feuille de travaux pratiques n° 1*
# Configuration et déploiement

*L'application développée ici sera réutilisée durant plusieurs séances de travaux pratiques. Il est donc important d'en garder une copie pendant toute la durée du module.*

On souhaite écrire une application de calcul de polynômes de la forme :

$$p(x) = \sum_{i=0}^{n} a_i x^i \quad \text{avec } a_n \neq 0$$

Le langage utilisé pourra être C ou C++ au choix. *La programmation devra être la plus modulaire possible.*

1. Écrire le programme demandant à l'utilisateur le degré du polynôme $p$ considéré et la liste de ses coefficients, ainsi que la valeur de $x$, puis affichant le résultat $s$ de l'évaluation de ce polynôme. La représentation en mémoire d'un polynôme se fera par un tableau des coefficients ou une liste chaînée des coefficients non nuls (le choix de la représentation sera déterminé grâce à une variable du fichier `Makefile`. Chaque binôme doit donc développer les deux représentations). Par ailleurs, l'évaluation des puissances se fera naïvement avec la fonction `pow()`. On utilisera un fichier `Makefile` sans règle implicite pour effectuer la compilation ;
2. Écrire un fichier `Makefile` minimal (i.e., utilisant le maximum de règles implicites) pour compiler l'application. On s'assurera qu'il est facilement possible de modifier les options de compilation (niveau d'optimisation du code, présence d'instructions de débogage, ...) ;

Dans les questions suivantes, on va utiliser les autotools et non plus seulement la commande `make`.

3. Écrire les fichiers `configure.ac` et `Makefile.in` minimaux pour l'application développée au point précédent ;
4. Ajouter une option `--with-float` compilant tout le programme avec des `float`s au lieu de `double`s ;
5. Ajouter le test vérifiant que la fonction `pow()` du système retourne bien 1 pour $0^0$. Si ce n'est pas le cas, le programme devra définir une fonction alternative avec ce comportement ;
6. On rappelle que les réels au format IEEE 754 `double` sont codés en machine sur 64 bits au format (signe:1, exposant+127:11, partie fractionnaire:52). Ajouter l'option `--enable-hexa` affichant la valeur de l'évaluation $s$ sous la forme (signe, exposant, partie fractionnaire) en hexadécimal. **Note :** on fera attention à assurer la portabilité de la routine d'affichage sur les ordinateurs gros- et petits-boutiens. De plus, l'utilisation de cette option devra forcer `--without-float` ;
7. Ajouter une option `--with-lists` ayant pour effet d'utiliser une représentation des polynômes par liste chaînée des coefficients non nuls. L'option `--without-lists` correspond à l'utilisation de la représentation à l'aide d'un tableau de coefficients (choix par défaut) ;
8. Modifier les fichiers de configuration pour utiliser `automake` (remplacement du fichier `Makefile.in` par un fichier `Makefile.am`) ;
9. Faire de la partie de l'application correspondant à la représentation des polynômes une librairie partagée à intégrer dans l'application principale avec *libtool* ;
10. Installer localement la *GNU Scientific Library* (GSL) disponible sur *madoc* ;
11. Modifier l'application pour pouvoir utiliser au choix de l'utilisateur (à spécifier lors de la configuration par l'option `--with-gsl`) la représentation des polynômes programmée précédemment ou celle offerte par la GSL (il faudra tester, le cas échéant, si la GSL est effectivement disponible) ;

12. Internationaliser l'application avec `gettext` ;
13. Ajouter une option `--with-gnulib` à `configure.ac` permettant d'utiliser les facilités de GNUlib en lieu et place des fonctions écrites par les développeurs précédents ou par vous (en particulier en ce qui concerne les listes chaînées). La documentation de GNUlib et la description des modules disponibles se trouvent sur *madoc*.