

# Plan de formation IDRIS

## Année 2011

8 mars 2011

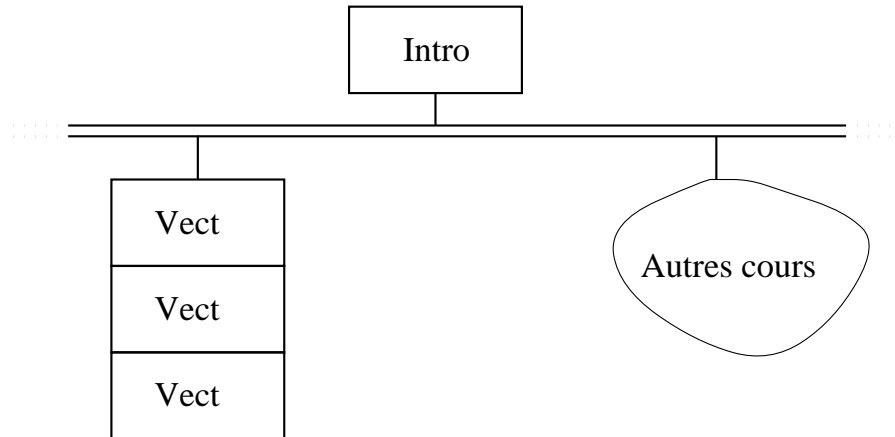
### Informations générales : modalités d'inscription.

- Certains<sup>1</sup> des cours décrits ici sont aussi publiés dans le catalogue de CNRSFormation<sup>2</sup>.
- **Pour l'inscription aux cours, adressez-vous :**
  - à Cours IDRIS
  - à CNRSFormation si vous n'appartenez pas au CNRS ou à l'Éducation Nationale.
  - Note : lorsque des cours sont organisés **en province**, ce sont les organismes locaux qui gèrent les inscriptions.
- Les inscriptions sont gérées et coordonnées globalement par l'IDRIS. Elles sont acceptées dans la limite des places disponibles et les cours peuvent être annulés si le nombre d'inscriptions est insuffisant. Huit jours (minimum) avant le début du cours, vous recevrez une confirmation d'inscription. Toute annulation doit être signalée et confirmée par écrit le plus tôt possible de façon à résorber au mieux la liste d'attente.
- Les cours peuvent être faits en province si le nombre d'inscriptions le justifie. Des formules condensées "à la carte" peuvent aussi être organisées.
- Les responsables de cours (cf. fiches ci-après) peuvent être joints par messagerie à l'adresse <Nom@idris.fr>.
- Les dernières mises à jour de ce plan ainsi que les documents relatifs aux cours peuvent être consultés via le serveur de l'IDRIS : [http://www.idris.fr/su/Cours/corps-Page\\_generale.html](http://www.idris.fr/su/Cours/corps-Page_generale.html)
- Pour ceux venant de province, des possibilités d'hébergement existent au centre de formation<sup>2</sup> de Gif-sur-Yvette à proximité de l'IDRIS.

<sup>1</sup>Cours C, F95-1, F95-2, Fortran 2003, MPI, Utilisation Blue Gene/P et OpenMP.

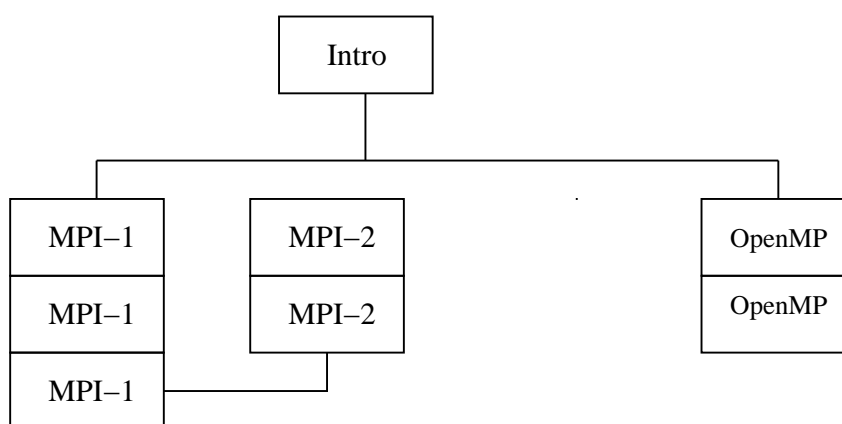
<sup>2</sup>CNRSFormation : bât. 31, av. de la Terrasse 91198 Gif-sur-Yvette Cedex Tél. 01-69-82-44-55 Fax 01-69-82-44-89  
<http://www.cnrsformation.cnrs-gif.fr>

<b>Cours : modules vectoriels</b>	<i>Machines</i>	<i>Durée jours</i>	<i>Nom</i>	<i>Cf. fiche page</i>	<i>Dates</i>
<b>Calcul vectoriel</b> : système de compilation, optimisation, bibliothèques, utilitaires d'analyse et de débogage.	SX-8	3	Vect	8	<i>à la demande</i>



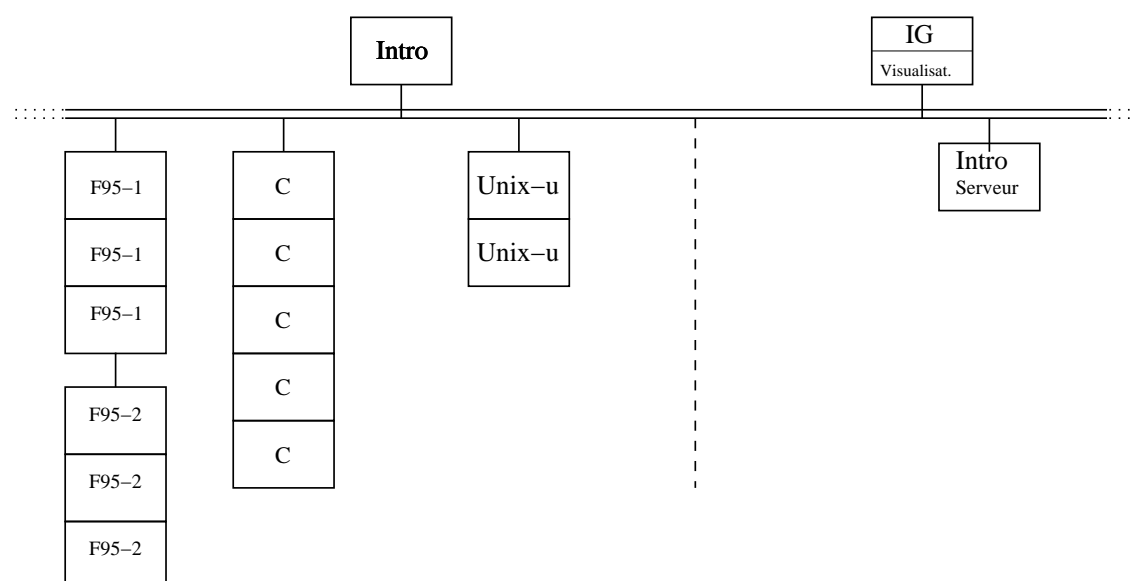
**Schéma : organisation des filières des modules vectoriels**

<b>Cours : modules parallèles</b>	<i>Machines</i>	<i>Durée jours</i>	<i>Nom</i>	<i>Cf. fiche page</i>	<i>Dates</i>
<b>Calcul parallèle : OpenMP</b>	Toutes	2	OpenMP	9	22-23/03/2011, 29-30/11/2011
<b>Calcul parallèle : MPI</b>	Toutes	2	MPI	10	31/01 au 02/02/2011, 09-12/05/2011, 10-13/10/2011
<b>Calcul parallèle : Hybride MPI/OpenMP</b>	Toutes	4	Hybride MPI/OpenMP	11	02-05/05/2011



**Schéma : organisation des filières des modules parallèles**

<b>Cours : modules généraux</b>	<i>Machines</i>	<i>Durée jours</i>	<i>Nom</i>	<i>Cf. fiche page</i>	<i>Dates</i>
<b>Utilisation de la machine Blue GeneP</b>	Toutes	2	Utilisation Blue Gene/P	12	08-09/02/2011, 27-28/09/2011
<b>Fortran 95-1</b>	Toutes	3	F95-1	13	25-27/01/2011, 05-07/04/2011, 04-06/10/2011
<b>Fortran 95-2</b>	Toutes	3.5	F95-2	14	15-18/03/2011, 17-20/05/2011, 15-18/11/2011
<b>Fortran 2003</b>	Toutes	3	Fortran 2003	15	21-23/06/2011
<b>Langage C</b>	Toutes	5	C	16	17-21/10/2011
<b>C++ et le calcul scientifique</b>	Toutes	2	C++	17	<i>à programmer ultérieurement</i>
<b>C++ et le calcul parallèle</b>	Toutes	2	C++ et le calcul parallèle	20	28-29/03/2011
<b>Unix : utilisation</b>	Toutes	2	Unix-u	21	<i>à la demande</i>
<b>Visualisation : module Introduction/Serveur</b>	Ulam	2	IG	22	<i>à la demande</i>



### Calendrier (janvier à juin 2011)

Schéma : organisation des filières des modules généraux

### Janvier 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
					1	2
3	4	5	6	7	8	9
10	11	12	13 C++ mod. D	14 C++ mod. D	15	16
17	18	19	20	21	22	23
24	25 F95-1	26 F95-1	27 F95-1	28	29	30
31 MPI						

### Février 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
	1 MPI	2 MPI	3 MPI	4	5	6
7	8 U_Babel	9 U_Babel	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28						

### Mars 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15 F95-2	16 F95-2	17 F95-2	18 F95-2	19	20
21	22 OpenMP	23 OpenMP	24	25	26	27
28	29	30	31			

### Avril 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
				1	2	3
4	5 F95-1	6 F95-1	7 F95-1	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

### Mai 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
						1
2	3	4	5	6	7	8
9 MPI	10 MPI	11 MPI	12 MPI	13	14	15
16	17 F95-2	18 F95-2	19 F95-2	20 F95-2	21	22
23	24	25	26	27	28	29
30	31					

### Juin 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21 Fortr. 2003	22 Fortr. 2003	23 Fortr. 2003	24	25	26
27	28	29	30			

## Calendrier (septembre à décembre 2011)

### Septembre 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27 U_Babel	28 U_Babel	29	30		

### Octobre 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
					1	2
3	4 F95-1	5 F95-1	6 F95-1	7	8	9
10 MPI	11 MPI	12 MPI	13 MPI	14	15	16
17 c	18 c	19 c	20 c	21 c	22	23
24	25	26	27	28	29	30
31						

### Novembre 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15 F95-2	16 F95-2	17 F95-2	18 F95-2	19	20
21	22	23	24	25	26	27
28	29 OpenMP	30 OpenMP				

### Décembre 2011

Lun	Mar	Mer	Jeu	Ven	Sam	Dim
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

**Nom** : Vect

**Durée** : 3 jours.

**Assistance maximale** : 20 personnes, **minimale** : 10 personnes.

### Plan

**Matinée du 1<sup>er</sup> jour** : système de compilation Fortran et bibliothèques scientifiques :

- principales options et directives du système de compilation ;
- contrôle de la compilation, de l'édition de liens et de l'exécution ;
- exécution en traitement par lots ;
- bibliothèques *propriétaires*, bibliothèques IMSL et NAG.

**Après-midi du 1<sup>er</sup> jour** : introduction à l'outil général Psuite et démonstrations :

- de compilation, d'édition de liens, d'exécution et de débogage ;
- de conversion Fortran 77  $\implies$  Fortran 90.
- Travaux pratiques.

**2<sup>e</sup> jour** : vectorisation et techniques d'optimisation :

- principes ;
- conflits de dépendance et conflits mémoire ;
- techniques d'optimisation et particularités du SX-8.
- Travaux pratiques.

**3<sup>e</sup> jour** : outils d'analyse et de performances :

- présentation des fonctions Fortran ou C ;
- présentation des commandes et des variables d'environnement ;
- présentation des outils X-Window disponibles sous Psuite.
- Travaux pratiques.

**Intervenants** : Jean-Michel DUPAYS, Patrick CORDE, Pierre-François LAVALLÉE.

## Calcul parallèle : OpenMP – *Open Multi-Processing*

**Nom** : OpenMP

**Responsable** : Pierre-François LAVALLÉE

**Objectif** : mise en pratique immédiate d'OpenMP grâce à une approche par l'exemple. Les nombreux schémas contenus dans ce cours, appuyés par une explication orale détaillée, montreront clairement les concepts inhérents à ce mode de parallélisation relativement efficace sur des machines multi-processeurs à mémoire partagée.

**Public concerné** : toute personne souhaitant paralléliser une application pré-existante ou dans sa genèse pour une architecture multi-processeurs à mémoire partagée.

**Pré-requis** : Fortran et Unix de base.

**Durée** : 2 jours.

**Assistance maximale** : 18 personnes.

### Plan

1. Introduction
2. Principes
3. Partage du travail
4. Synchronisation
5. Quelques pièges
6. Performances
7. Conclusion

Travaux pratiques : environ 50 % du temps total.

**Intervenants** : Pierre-François LAVALLÉE, Pascal VOURY.

## Calcul parallèle : MPI– *Message Passing Interface*

**Nom** : MPI

**Responsable** : Isabelle DUPAYS

**Objectif** : pouvoir développer des programmes parallèles de par la mise en oeuvre de la bibliothèque d'échange de messages MPI.

**Public concerné** : tout utilisateur désirant utiliser MPI sur des plates-formes parallèles.

**Pré-requis nécessaires** : Pour les travaux pratiques, connaissances de base d'Unix et du langage Fortran.

**Durée** : 4 jours.

**Assistance maximale** : 20 personnes.

### Plan

#### 1<sup>er</sup> jour

- Introduction,
- Environnement de MPI (avec exercice),
- Communications point à point (avec exercice),
- Communications collectives (avec exercice).

#### 2<sup>e</sup> jour

- Types dérivés (avec exercice),
- Optimisations,
- Topologies.

#### 3<sup>e</sup> jour

- Communicateurs (avec exercice),
- Gestion de processus (1/2 journée).

#### 4<sup>e</sup> jour

- Entrées-sorties parallèles (avec exercice),
- Exercice récapitulatif (1/2 journée).

**Intervenants** : Isabelle DUPAYS, Dimitri LECAS, Pierre-François LAVALLEE, Philippe WAUTELET.

**Nom** : Hybride MPI/OPenMP

**Responsable** : Philippe WAUTELET

**Objectif** : L'objectif principal de cette formation sera d'initier les participants à la programmation hybride MPI/OpenMP. Ce type de programme est particulièrement bien adapté pour l'exploitation optimale des super-calculateurs, notamment ceux des centres nationaux. Cette formation permettra de présenter les concepts sous-jacents, de mettre en avant les différents intérêts de cette approche (performance, extensibilité, optimisation de la mémoire...), de décrire les différentes implémentations possibles et de les mettre en oeuvre explicitement sur une application réelle d'hydrodynamique.

**Public concerné** : ingénieurs et chercheurs.

**Pré-requis** :

- Connaissance et utilisation de Fortran 90/95 ou C,
- Connaissance de base de la bibliothèque échange de message MPI,
- Connaissance de base d'OpenMP.

**Durée** : 4 jours.

**Assistance maximale** : 20 personnes,

## Plan

- **Introduction/problématique**
- **MPI avancée (1 jour)**
  - Facteurs influençant les performances MPI,
  - Modes d'envoi des communications point à point et comparaison avec les communications collectives,
  - Recouvrement des calculs par les communications,
  - Types dérivés,
  - Problème de l'équilibrage de charge,
  - Placement des processus.
- **OpenMP avancé (1 jour)**
  - Limitations d'OpenMP,
  - Approche classique *Fine Grain* (FG),
  - Approche *Coarse Grain* (CG),
  - Performances comparées MPI/OpenMP-FG/OpenMP-CG.
- **Hybride MPI/OpenMP (2 jours)**
  - Motivations,
  - Présentation des concepts généraux,
  - MPI et OpenMP,
  - Adéquation à l'architecture : l'aspect gain mémoire,
  - Adéquation à l'architecture : l'aspect réseau,
  - Etudes de cas,
  - Outils.

70% du temps sera consacré aux TP (langage C et Fortran90). Machines cibles de l'IDRIS : Vargas (IBM Power6, 3584 coeurs) et Babel (IBM BG/P, 40960 coeurs)

**Intervenants** : Philippe WAUTELET, Pierre-François LAVALLÉE.

**Nom** : Utilisation Blue Gene/P

**Responsable** : Philippe WAUTELET

**Objectif** : acquérir une vue d'ensemble des caractéristiques de la machine Blue Gene/P de l'IDRIS, de son utilisation et de sa programmation.

**Public concerné** : toutes les personnes désirant travailler sur cette machine.

**Pré-requis** : connaissance de base de Fortran 95, de MPI, de systèmes Unix/Linux et d'un éditeur comme vi, emacs ou xedit).

**Durée** : 2 jours.

**Assistance maximale** : 15 personnes, **minimale** : 10 personnes.

## Plan

- **Introduction**
- **Hardware**
  - Structure Blue Gene/P
  - Configuration complète (BG/P + Power6)
  - Coeurs/noeuds de calcul
  - Double FPU
  - Réseaux
  - Entrées/sorties
- **Software**
  - Systèmes
  - Infrastructure de communications
  - MPI
  - Mapping/placement des processus
  - Multithreading
  - compilation
  - Double FPU
  - Soumission
  - Modes d'exécution
  - Bibliothèques
  - Applicatifs
  - Outils
  - addr2line
  - MPItrace
  - GNU gprof

Avec Travaux pratiques.

**Intervenants** : Philippe WAUTELET.

Fortran 95-1  
(Fortran : notions de base)

**Nom** : F95-1

**Responsable** : Patrick CORDE

**Objectif** : acquérir et mettre en pratique les structures de base du langage Fortran. Une fois consolidées par la pratique, ces notions permettent alors d'aborder avec profit le cours Fortran F95-2

**Public concerné** : toutes les personnes désirant développer des codes scientifiques.

**Pré-requis** : bases Unix, pratique d'un éditeur et notions de programmation et d'algorithmique.

**Durée** : 3 jours.

**Assistance maximale** : 20 personnes, **minimale** : 10 personnes.

### Plan

1. Introduction (historique, bibliographie et documentation, ...)
2. Généralités (bases de numération, représentation des données, jeux de caractères ...)
3. Déclarations (types de base , IMPLICIT NONE, EQUIVALENCE, ...)
4. Opérateurs et expressions
5. Structures de contrôle : tests, boucles ...)
6. Tableaux (déclarations, instruction DATA, ...)
7. Entrées-Sorties (fichiers binaires, formatés, ...)
8. Procédures (Subroutine, Function, Procédures intrinsèques)
9. Common (BLOCK DATA, SAVE, ...)

Avec travaux pratiques.

**Intervenants** : Romuald CARPENTIER, Patrick CORDE.

Fortran 95-2  
(apports des nouvelles normes)

**Nom** : F95-2

**Responsable** : Patrick CORDE

**Objectif** : acquérir l'ensemble des nouveautés des normes Fortran 90 et 95 avec mise en pratique des concepts.

**Public concerné** : toutes les personnes désirant programmer en Fortran 90/95.

**Pré-requis** : bonne connaissance de Fortran 77, des bases Unix et d'un éditeur. À défaut, il est recommandé d'assister préalablement au cours Fortran 95-1.

**Durée** : 3 jours.

**Assistance maximale** : 20 personnes, **minimale** : 10 personnes.

### Plan

1. Introduction (historique, bibliographie, documentation, ...)
2. Généralités (éléments syntaxiques, structure d'un programme, ...)
3. Structures de données : types dérivés
4. Structures de contrôle : programmation structurée
5. Extensions tableaux
6. Gestion mémoire
7. Pointeurs
8. Interface de procédures et modules (surcharge d'opérateurs, ...)
9. Contrôle de visibilité, concept d'encapsulation et gestion des zones dynamiques inaccessibles en retour de fonction.
10. Procédures récursives
11. Nouveautés sur les E./S. et nouvelles fonctions intrinsèques

Avec travaux pratiques.

**Intervenants** : Jean-Michel DUPAYS, Patrick CORDE.

## Fortran 2003

**Nom** : Fortran 2003

**Responsable** : Patrick CORDE

**Objectif** : acquérir l'ensemble des nouveautés de la norme Fortran 2003 avec mise en pratique des concepts.

**Public concerné** : toutes les personnes désirant tirer profit des nouveaux concepts de Fortran/2003 tels que les concepts objets.

**Pré-requis** : La maîtrise du Fortran/90 est impérative : avoir suivi la formation F95-2 ou avoir les connaissances équivalentes.

**Durée** : 3 jours.

**Assistance maximale** : 20 personnes, **minimale** : 10 personnes.

### Plan

1. Nouveautés concernant les tableaux dynamiques (attribut ALLOCATABLE),
2. Nouveautés concernant les modules (attributs PROTECTED, IMPORT).
3. Nouveautés concernant les entrées-sorties :
4. nouveaux paramètres des instructions OPEN/READ/WRITE,
5. Méthode d'accès STREAM,
6. entrées-sorties asynchrones.
7. Pointeurs de procédures,
8. Paramétrage des types dérivés,
9. Extension d'un type dérivé,
10. Variable polymorphique,
11. Procédures attachées à un type dérivé,
12. Héritage,
13. Type dérivé abstrait,
14. Traitement personnalisé des objets de type dérivé au sein d'entrées-sorties,
15. Interfaçage C-Fortran,
16. Arithmétique IEEE et traitement des exceptions.

Avec travaux pratiques.

**Intervenants** : Patrick CORDE.

## Langage C

**Nom** : C

**Responsable** : Patrick CORDE

**Objectif** : acquérir et mettre en pratique les éléments de base ainsi que les méthodes de la programmation en langage C.

**Public concerné** : toute personne désirant apprendre à programmer en langage C.

**Pré-requis** : bonne connaissance d'un langage de programmation, des bases Unix et d'un éditeur.

**Durée** : 5 jours.

**Assistance maximale** : 20 personnes, **minimale** : 10 personnes.

### Plan

1. Présentation du langage
2. Concepts fondamentaux
3. Structures et unions
4. Opérateurs et expressions
5. Pointeurs
6. Tableaux
7. Fonctions
8. Durée de vie et portée des identificateurs
9. Manipulation d'adresses
10. Préprocesseur
11. Bibliothèque standard

Avec travaux pratiques.

**Intervenants** : Jean-Michel DUPAYS, Patrick CORDE.

## C++

**Nom** : C++

**Responsable** : Victor ALESSANDRINI

**Objectif** : une expérience pédagogique dans la programmation orientée objet et la programmation générique, orientée vers le calcul scientifique.

**Public concerné** : toute personne désirant apprendre à programmer en langage C++.

**Durée** : 4 modules de 2 jours chacun.

**Assistance maximale** : 25 personnes, **Assistance minimale** : 15 personnes.

### Plan

Une expérience pédagogique dans la programmation orientée objet et la programmation générique, orientée vers le calcul scientifique.

Cette note se propose de préciser les motivations, les objectifs et les modalités adoptées pour la mise en place - à titre d'expérience initiale - d'une série de formations C++ qui se tiendront à l'IDRIS dans les mois qui viennent.

Cette formation s'adresse naturellement aux scientifiques et ingénieurs utilisateurs des centres nationaux de calcul. Des membres des organisations industrielles partenaires de la recherche publique seront accueillis dans la mesure de la disponibilité des places.

A - La pertinence de C++

C++ est une extension de C conçue pour supporter un style de programmation basé sur l'encapsulation des données, la programmation orientée objet ainsi que la programmation dite « générique », qui permet d'écrire des « patrons de codes » (templates) qui s'adaptent à des types de données différents. Il va de soi que la totalité du langage C (y compris la bibliothèque standard C) est préservée et reste disponible pour le programmeur, mais C++ pousse les frontières dans plusieurs directions qui facilitent énormément le développement de codes complexes. C++ propose au programmeur un nombre important de nouveaux outils et de stratégies de programmation, mais il ne force pas leur adoption. Il est ainsi possible (et même recommandé) de suivre un apprentissage incrémental de C++.

Presque trois décennies après l'introduction du langage, la programmation orientée objet n'est plus une nouveauté : elle est adoptée par Java, C#, Python, toute une série de langages de programmation qui font l'unanimité dans le monde non-scientifique, c'est-à-dire la presque totalité des programmeurs de la planète. Java, qui recouvre C++ à environ 80

La bibliothèque standard C++ inclut la « Standard Template Library » (STL), une pièce brillante de génie logiciel basée sur la programmation générique qui facilite énormément le développement des codes demandant une manipulation des données un tant soit peu sophistiquée. Le style de programmation apporté par la STL est très répandu. Citons à titre d'exemple une extension ad-hoc de la STL adaptée à la programmation CUDA des GPGPUs. Il est par ailleurs possible de profiter de la puissance expressive de C++ pour simplifier le développement de bibliothèques, et de les encapsuler par la suite derrière des interfaces C, utilisables par un code C (sans doute) ou Fortran (peut-être).

Pour ce qui concerne le calcul scientifique, il existe un certain nombre de bibliothèques C++ qui utilisent les technologies du langage pour proposer des codes performants pour certains problèmes scientifiques. Mais, d'une manière générale, les abstractions sémantiques de C++ sont à manipuler avec prudence pour les noyaux de calcul où la performance est critique : il vaut mieux ne pas trop s'éloigner de la sémantique traditionnelle C. En revanche, la plus-value apportée par C++ se trouve :

1. Dans la structuration et l'organisation des codes. C++ va bien au-delà de la programmation structurée traditionnelle, en proposant une panoplie de méthodes pour rendre les codes complexes plus souples, réutilisables et extensibles. Ceci est le coeur de la programmation orientée objet.
2. Dans la simplification du développement des codes apportée par l'encapsulation des données dans des objets, et par l'utilisation de la programmation générique (en particulier la STL).

Les applications scientifiques devenant de plus en plus sophistiquées avec la complexité des simulations qui sont engagées, la question de la structuration et de l'organisation des codes prend une place de plus en plus importante.

#### B - Les objectifs de la formation

Il est important de reconnaître que C++ - comme Java, d'ailleurs - n'est pas juste un langage avec une syntaxe. C'est aussi un environnement puissant de génie logiciel.

La seule connaissance de la syntaxe C++ ne suffit pas pour programmer avec efficacité. Le programmeur doit connaître les forces et les faiblesses de l'ensemble d'options qui lui sont offertes. Il doit connaître également un certain nombre de styles de programmation - les « design patterns » - qui sont des standards « de facto » de génie logiciel ayant prouvé leur efficacité pour résoudre des problèmes spécifiques qui se présentent dans la programmation courante.

Dans la formation qui est proposée, un effort a été fait pour présenter les sujets techniques de manière à aborder non seulement les comment mais aussi les pourquoi. Une réflexion sur l'impact des techniques de programmation C++ dans les stratégies de développement des codes est sous-jacente dans la presque totalité des sujets abordés.

La formation s'appuie sur une grande quantité d'exemples de trois types (tous des codes complets, et disponibles, à compiler et exécuter en séance) :

- Exemples simples qui démontrent l'utilisation d'une syntaxe particulière. Un ou plusieurs exemples simples sont disponibles pour chaque sujet abordé.
- Codes d'une bibliothèque utilitaire qui fournissent des services aux applications clientes : optimisation de l'allocation de mémoire, gestion simplifiée des entrées-sorties, gestion des transactions, points de reprise (checkpoint-restart), génération de nombres aléatoires, multithreading, etc. Ces codes illustrent les modalités de développement de codes en C++.
- Quelques applications complètes pour illustrer les stratégies de développement des applications un tant soit peu complexes en C++. Une application de dynamique moléculaire est utilisée pour illustrer un certain nombre de « design patterns » disponibles dans la programmation orientée objet. Un paquetage pour la solution d'un système d'équations différentielles non-linéaires de premier ordre est utilisé pour illustrer toutes les possibilités du « polymorphisme dynamique » (programmation objet) et du « polymorphisme statique » (programmation générique).

Les sources de tous ces exemples sont abondamment commentées.

#### C - Le contenu et le support de la formation

Les objectifs ayant été précisés, nous avons essayé de définir une manière efficace de les atteindre. L'ensemble de la formation C++ sera initialement proposée sous la forme de quatre modules se suivant dans un ordre logique, dans la mesure où les connaissances développées dans un module sont nécessaires pour les suivants. La durée estimée de chaque module est d'environ 2 jours

Module A : De C à C++ : toutes les extensions C++ à C et à la programmation procédurale traditionnelle. Discussion de la programmation avec des objets, et introduction à la programmation générique.

Module B : L'utilisation de la bibliothèque standard C++, incluant la bibliothèque « iostream » pour les entrées-sorties et la « Standard Template Library » (STL).

Module C : La programmation orientée objet comme un ensemble de techniques de génie logiciel destinées au développement d'applications flexibles et facilement extensibles (polymorphisme dynamique). Programmation générique avancée avec le même objectif (polymorphisme statique). « Design patterns » et autres stratégies de développement de codes

Module D : C++ et le calcul parallèle à mémoire partagée (systèmes multi-coeurs) : introduction à la programmation avec des « threads », bibliothèques de multithreading qui utilisent fortement la sémantique C++ (comme Threading Building Blocks d'Intel) et extensions parallèles simples du langage (Cilk++).

En plus d'une présentation PowerPoint d'environ 100 diapositives pour chaque module et des nombreux exemples qui vont avec, un certain nombre de documents annexes accompagnent et amplifient les contenus des présentations PowerPoint, afin de faciliter leur apprentissage. Il va de soi que ces documents - comme les présentations PowerPoint - font référence aux exemples utilisés dans le cours. A la date d'aujourd'hui, deux documents (rédigés en anglais) sont disponibles :

1. Notes on C++ I/O (60 pages) VA, 2010
2. Notes on the Standard Template Library (100 pages) VA, 2010

Cette organisation est naturellement susceptible d'évoluer en fonction de l'expérience acquise lors de sa mise en place.

Victor Alessandrini

D - Agenda

Module A : 21-22 septembre 2010

Module B : 09-10 novembre 2010

Module C : 30/11 au 01 décembre 2010

**Intervenants** : Victor ALESSANDRINI,

## Langage C++ et le calcul parallèle

**Nom :** Langage C++ et le calcul parallèle

**Responsable :** Victor ALESSANDRINI

**Objectif :** Cette formation se propose de présenter un certain nombre d'environnements de programmation et de bibliothèques pour le calcul parallèle à mémoire partagée (multithreading) disponibles dans un environnement C++ : threads natifs, OpenMP, ainsi que les modèles parallèles d'Intel (Threading Building Blocks, Cilk++, Array Building Blocks). On discutera en particulier leur interoperabilité et leur complémentarité. Le cours est plus axé sur la programmation parallèle que sur C++, qui se singularise parce que les très intéressants modèles parallèles d'Intel ne sont disponibles que dans ce langage. Des connaissances de base de C++ sont amplement suffisantes. Seule la bibliothèque Threading Building Blocks demande une connaissance un peu plus approfondie de la programmation orientée objet.

Les exercices simples à faire en cours et les exemples plus complexes sont déclinés dans tous les environnements et bibliothèques pour faciliter leur comparaison.

**Public concerné :** toute personne désirant utiliser la programmation parallèle.

**Durée :** 2 jours.

**Assistance maximale :** 25 personnes, **Assistance minimale :** 15 personnes.

### Plan

1. Introduction au calcul parallèle à mémoire partagée,
2. Généralités sur les threads,
3. Généralités sur la synchronisation des threads,
4. Classes C++ de synchronisation et de management de threads (bibliothèque « vath »),
5. OpenMP en environnement C++,
6. Threading Building Blocks,
7. Cilk++,
8. Array Building Blocks.

**Intervenants :** Victor ALESSANDRINI,

## UNIX : utilisation

**Nom** : Unix-u

**Responsable** : Pascal VOURY

**Objectif** : fournir les connaissances de base du système Unix nécessaires pour aborder efficacement le développement et l'exploitation de programmes dans un environnement scientifique.

**Public concerné** : tout utilisateur travaillant sous un système Unix et souhaitant améliorer son efficacité.

**Pré-requis** : maîtrise minimale d'un environnement utilisateur Unix.

**Durée** : 2 jours.

**Assistance maximale** : 20 personnes, **minimale** : 8 personnes.

### Plan

1. Pourquoi se mettre à Unix ?
2. Le démarrage, un mystère !
3. Où trouver l'information ?
4. Tout faire avec et grâce aux fichiers.
5. Je cherche un fichier et je me perds dans tous ces répertoires !
6. Comment communiquer simplement son travail à quelqu'un d'autre ?
7. L'éditeur et le contrôle de la ligne de commande.
8. La gestion des flux et des processus.
9. Ne pas faire n fois le même travail !
10. Travailler agréablement...

**Intervenants** : Pascal VOURY, Raphaël MEDEIROS.

## Visualisation : module Introduction/Serveur

**Nom** : IG

**Responsable** : Thierry GOLDMANN

**Objectif** : faire un tour d'horizon de la visualisation de données, avoir une idée des logiciels et matériels disponibles à l'IDRIS pour cela.

**Public concerné** : toute personne désirant interpréter ses résultats de calcul.

**Pré-requis** : connaissances de base sur Unix et l'environnement X. Pratique de Fortran ou C.

**Durée** : 2 jours.

**Assistance maximale** : 12 personnes, **minimale** : 5 personnes.

### Plan

• **1<sup>er</sup> jour** :

- tour d'horizon et prospective,
- logiciels disponibles à l'IDRIS,
- le serveur de visualisation : Ulam,
- la vidéo à l'IDRIS,
- aperçu de l'utilisation d'AVS en réparti.

• **2<sup>e</sup> jour** :

atelier personnalisé avec les utilisateurs pour les aider à démarrer, à trouver avec eux l'outil et le mode de travail le mieux adapté à leurs besoins, leurs données.

**Intervenants** : Marc RUGERI, Sylvie THEROND, Thierry GOLDMANN.