# Comment ne pas progresser en programmation*



## 1 Tout à propos de la programmation, au sens le plus strict du mot

### 1.1 Ignorer les messages

Les compilateurs, les systèmes d'exploitation, ... génèrent des messages d'erreur uniquement conçus pour être lus par leurs auteurs (peut-être pour justifier leurs salaires). Un temps précieux est gaspillé par la lecture de ces messages, ce temps pourrait être mieux dépensé... écrire du code, bien sûr ! Les messages d'erreur nous rendent moins productifs. Ne tombez pas dans le piège. Ignorez les.

Idem pour les messages d'avertissement, les ignorer vous permettra de montrer que vous êtes un programmeur expérimenté qui n'a pas peur des ordinateurs. Quelle meilleure manière de montrer son expérience en tant que programmeur que de distribuer un programme qui génère des dizaines, non, des centaines de messages d'avertissement ? Tout le monde peut voir que vous êtes expérimenté, juste un programmeur trop occupé pour perdre du temps sur de tels messages.

### 1.2 Ne pas arrêter de penser

Ne vous méprenez pas ici. Qu'allons nous faire ? Un programme. Quelle est la seule chose qui compte vraiment dans un programme ? Le code. Qu'est ce qui fonctionne vraiment ? Le code. Pourquoi utiliser des ressources dépassées comme des crayons, des stylos ou du papier ? Vous faites partie de la génération SMS, vous n'êtes pas fou pour écrire des choses simples, consommatrices de temps, non ? Ensuite, arrêtez de ne penser à rien alors qu'il y a tant de choses à faire : du code.

Vous ne devez jamais arrêter le codage. Nous savons tous que les messages d'erreur sont d'inacceptables interruptions, un obstacle inutile, alors que nous nous concentrons sur notre sujet. Que faites-vous si vous obtenez un message d'erreur de compilation ? Comme vous devriez le savoir maintenant, la lecture de ce message et sa compréhension ne sont tout simplement que des options.

Vous pouvez essayer de faire quelques changements aléatoires dans le code source. On ne sait jamais, vous pourriez jeter de la poudre au yeux du compilateur. Mais si cela ne fonctionne pas, ne perdez pas davantage de temps. NON, ne soyez pas tentés d'essayer de lire le message ou de le comprendre. Simplement concentrez vous sur la production du code ; c'est la seule façon de finir cette horrible mission. Comme nous le savons tous, les erreurs ont tendance à disparaître par elles-mêmes. À la fin de la journée, vous compilez, vous lancez le programme ; et même si vous aviez testé (non pas que vous en aviez besoin) vous aviez vu que tout était OK (les erreurs pouvant disparaître d'elles-mêmes).

Si le code compile mais que quelque chose ne va pas, cela n'importe pas ; résolvez ce problème plus tard, quand vous aurez fini. Quoi qu'il en soit, vous pourriez avoir de la chance et apprendre que les chargés de cours ont changé l'objectif et que cela s'inscrit dans le cadre de votre programme. Ne prenez donc pas le risque de corriger des programmes ; vous pourriez perdre votre temps.

*Document humoristique, un brun ironique, un tantinet provocateur

### 1.3 Je ne veux pas de problème

Si votre programme contient un bogue qui survient chaque fois, il sera difficile à trouver, mais il n'apparaîtra pas lors de l'examen, de la démonstration, .... Peut-être qu'il disparaîtra tout seul. Ne vous inquiétez pas. Mais si le bug se présente encore et encore, changez les choses au hasard jusqu'à ce qu'il disparaisse. Si vous décidez de vous débarrasser du bogue - simplement parce que l'envie vous en prend ; il suffit d'écrire le même code de différentes manières. Peut-être que le problème disparaîtra ; clairement, il s'agit là de l'attitude la plus professionnelle.

Ne compilez pas régulièrement, n'entravez pas votre progression. Vous êtes un professionnel et un professionnel programme rapidement. Écrivez des milliers de lignes de code d'abord, laissez la compilation de côté, il sera beaucoup plus divertissant et intéressant de rechercher les erreurs de compilation.

La même règle s'applique pour les erreurs d'exécution. Si vous essayez de maintenir votre programme, de le corriger comme il se doit, il sera trop facile de repérer un nouveau bogue. Seuls les lâches les font. Un vrai programmeur écrit tout le programme, puis il digère son ensemble comme un boa constrictor. Recherchez un bug caché dans les dernières 10000 lignes est excitant, mais si il y a seulement 10 ou 20 lignes, quel plaisir y a t il à cela ?

Et pourquoi utiliser des débogueurs ? C'est à l'enseignant de chercher vos bogues. Les erreurs de programmation sont l'exception, non la règle, et quand vous deviendrez un professionnel, vous n'aurez plus à les corriger. Pourquoi perdre votre temps ou alors dépenser votre énergie à apprendre à les traiter dans le cadre de votre enseignement ?

## 2 Si seulement je pouvais trouver les mots

### 2.1 Commentaires

Les commentaires techniques et les spécification sont vraiment pénibles. Ces fastidieuses et longues explications présentent des problèmes non pertinents et ne sont que l'occasion pour les enseignants de présenter leurs traits narcissiques. Vous avez seulement besoin de jeter un œil sur eux. Les commentaires ne font pas partie de votre mission, qui n'est autre que ...l'écriture de code.

D'autre part, l'intérêt des règles de codage est de montrer comment nos enseignants sont arrogants. Ils aiment nous contrôler, nous obligeant à faire des exercices inutiles - c'est pourquoi ils rédigent des règles en premier lieu. Ne jouez pas leur jeu. La lecture ou l'application de ces règles ne peuvent pas rendre votre travail meilleur. Et pourquoi rendre vos exercices faciles à noter, ils sont payés pour les corriger, n'est-ce pas ? Ne prenez même pas la peine de mettre votre nom ou votre classe sur vos exercices. Les enseignants auront peu de difficultés à se souvenir de votre visage et de votre style de programmation incomparable, ils sauront que cela vient de vous en tout cas.

### 2.2 Rédaction

N'écrivez pas de commentaire. Nous l'avons dit auparavant et nous le répétons : quel est l'intérêt de tout cela ? Pour créer un programme, c'est-à-dire du code. Des morceaux non exécutables, inutiles ; les explications sont une insulte à l'intelligence d'un programmeur - après tout, il ou elle peut lire le code, non ?

S'il y a des commentaires obligatoires (les descriptions des fonctions et des choses comme ça), alors écrivez les, même si vous n'avez rien d'intéressant à dire. Les enseignants aiment ces balivernes, comme ça, vous obtiendrez de meilleures notes.

Quant à la documentation, écrivez la à la fin. Comment pouvez vous écrire un document décrivant un programme qui n'existe pas encore ? Quel est l'intérêt de rédiger des documents pour vous sur ce que vous venez de faire ? La seule raison de l'écriture de la documentation pour un programme, c'est que les professeurs le demandent. C'est quelque chose que vous pouvez faire la journée avant la date limite. En plus, il n'y a aucune chance que vous oubliez tout ce que vous venez de faire.

Utilisez aussi des abréviations à consonance étrange. Les enseignants sont tellement vieux. Vous êtes un membre de la génération SMS. Essayez d'écrire des messages qui sont difficiles à

lire. L'enseignant devra faire un effort supplémentaire, après une longue journée coincé devant un écran d'ordinateur. Autant d'éléments qui devraient contribuer à élever le niveau de concentration du vieux et de mettre celui-ci dans une véritable bonne humeur.

Qu'en est-il de l'orthographe ? L'orthographe est démodée. De toute évidence, vous avez le droit d'écrire comme vous le souhaitez.

Avouons-le, qui n'aime pas faire de fautes d'orthographe ? C'est trop facile. Vous voulez donner à votre professeur une gifle ? Donnez lui une ligne comme : «Je suis en train de faire l'ekzo pour vous. Je pense que ses difficiles.» Il comprendra, ne vous inquiétez pas.

# 3 Votre relation avec votre professeur

## 3.1 Ne pas demander de l'aide

S'il y a quelque chose que vous ne pouvez pas faire, si vous avez des questions ou si vous êtes perdu, ne cherchez pas de l'aide, ne posez pas de question au cours du cours et n'assistez pas aux TP. Il y a des milliers de raisons pour lesquelles vous ne devriez pas, en voici quelques unes :

- Se rendre aux TP, en posant des questions revient à admettre que vous êtes stupide.
- Mieux vaut être ignorant que de courir le risque de révéler que vous ne savez pas quelque chose que vous devriez.
- Quand vous posez une question au cours d'un cours, vos camarades vont penser que vous êtes stupide. Vous ne le pensez pas quand ils posent une question, mais ils le feront à votre sujet. Cet argument vaut pour chaque étudiant dans une salle de cours à un moment donné, c'est pourquoi aucun d'entre eux ne peut poser de questions.

Conclusion : ne jamais demander de l'aide ou aller à un TP. Il existe cependant une exception à cette règle, vous êtes autorisé à passer voir l'enseignant dans les derniers jours avant une date limite. Il peut y avoir une longue file d'attente, il va consacrer son temps à aider les élèves tout en négligeant d'autres tâches, mais ne vous inquiétez pas, il ne sera pas en mesure de résister à la tentation de vous aider dans cette sombre et lugubre heure de besoin.

## 3.2 Défiez vos enseignants

Si, malgré tout ce que nous avons dit, vous décidez de demander de l'aide, toujours se rappeler une règle d'or qui vous aidera dans votre carrière professionnelle - après tout une foule d'utilisateurs de l'ordinateur le font aussi. Ne JAMAIS donner une description détaillée du problème.

Voici un exemple. Si quelque chose de fâcheux survient alors que vous construisez un programme, allez voir l'enseignant et lui dire : «*Quelque chose de bizarre s'est produit avec mon programme d'hier.*». Il vous regarde, en attendant plus de détails. Mais ne cédez pas, ne dites pas quoi que ce soit d'autre. Ne pensez même pas entrer dans les détails tels que :

- L'étrange événement s'est produit lors de la compilation du programme ou pendant l'exécution.
- L'étrange incident a provoqué la fin soudaine du programme ou une boucle infinie, ou tout simplement, le programme n'a pas fait ce que vous aviez prévu.

Voici un autre exemple. Si l'étrange événement s'est produit lorsque vous compiliez, ne pas dire à l'enseignant ce que dit le message d'erreur ou de la ligne de code où il est apparu. Il suffit de dire quelque chose comme : «*Il a donné quelques messages d'erreur, ou autre.*»

Voici encore un autre exemple. Si l'étrange événement s'est produit au moment de l'exécution du programme et a provoqué une terminaison brutale, ne jamais donner le message d'erreur. Il suffit de dire : «*Il a donné quelques messages d'erreur, ou autre.*»

Un dernier exemple. Si l'étrange événement s'est produit après une modification, ne surtout pas dire ce que vous avez modifié. Il suffit de dire : «*Je ne comprend pas tout à l'heure ça marchait, mais maintenant ça ne marche plus.*»

Évitez les descriptions de ce type : «*Cette erreur se produit à chaque fois que je charge un deuxième fichier et que le premier était vide.*». Il s'agit de dire les mots magiques : «*Il a donné quelques messages d'erreur, ou autre.*»

Si vous persistez à vouloir être irresponsables et demandez de l'aide dans les TP, sans même

penser à cerner le problème avant d'y aller. S'il y a une erreur dans un fichier d'entrée de 1 Mo. Ne tentez pas avec des fichiers plus petits, jusqu'à ce que vous ayez identifié la cause de l'erreur. N'essayez pas de créer un mini programme. Si vous ne le faites pas, l'enseignant trouvera probablement le problème immédiatement. C'est le genre de défi qu'il adore ! Le mieux à faire est de lui donner à lire des milliers de lignes de code. Ça va lui donner la chance de pratiquer ses habiletés et vous serez en mesure de vérifier ses capacités de déduction.

### 3.3 Soyez malin en utilisant le courrier électronique

Certaines questions sont presque impossibles à répondre par e-mail. Entretenez cette compétence à répondre à vos questions. Voici un exemple : «*Il a donné quelques messages d'erreur, ou autre chose. J'ai joint le code source.*». Vous pouvez aller dans l'autre sens également, si vous le désirez, en posant une question plus spécifique, mais en oubliant d'envoyer le code : «*Le constructeur dans ma classe a donné quelques messages d'erreur, ou autre.*»

Il va sans dire que vous devriez écrire votre message et l'envoyer immédiatement. Ne jamais relire les messages. Il y a une autre raison pour laquelle les e-mails sont tellement agréable. Vous pouvez cacher votre nom, groupe, classe dans un pseudo. Tout sera OK si vous prenez l'approche informelle - cela rend le contact tellement plus intime, votre nom est un détail sans importance.

## 4 Sans oublier

### 4.1 Laissez tout pour la toute dernière minute

Dès le premier jour vos professeurs vous diront de travailler régulièrement pendant la semaine. Ne les écoutez pas.

Bien que cela puisse être une discipline relativement nouvelle, la programmation a déjà acquis un certain nombre de traditions sacrées, dont l'une est le rush de dernière minute pour obtenir votre travail dans le temps. Vous soumettre à ce stress est un élément essentiel de vous préparer pour le monde du travail. Relax. Laissez votre travail s'accumuler peu à peu et ignorer allègrement les avertissements et les signes révélateurs que vous êtes en retard. Ne laissez pas les études dominer votre style de vie. Ne vous laisser pas interrompre votre voyage par une vaine tentative de rattraper le temps perdu. Juste au moment où vous êtes au bord du précipice, juste au moment où vous n'avez plus que deux semaines devant vous pour faire un programme pour lequel vous avez eu quatre mois pour le faire, alors là codez.

Quel serait l'attraction de la programmation si nous n'avions pas à réaliser des programmes dans l'urgence ? Qu'en serait-il de l'image de la longue chevelure, la barbe, l'odeur nauséabonde d'un programmeur tapant sur un clavier à l'écart pendant 48 heures non-stop ? Auriez-vous l'endurance pour passer à la LAN Party, assis sur une chaise en plastique et passer trois jours à l'étroit dans un chapiteau à 35 degrés pour abattre les monstres sur un écran ? De quel droit aurions-nous le droit de nous appeler héros si nous faisions une pause chaque jour uniquement parce que nous sommes senti un peu fatigué ? Il suffit de penser : Que se passerait il pour Coca-Cola et de l'ensemble de ses usines ? Et que se passerait-il à l'ensemble des usines de café qui consacrent la moitié de leur production à des programmeurs informatiques ?

Être à jour dans votre travail et dans votre compréhension de ce qui se passe dans la salle de cours est bon pour les travailleurs et autres mauviettes. Vous vous savez ce qu'il faut faire.

## 5 Tricher

Copiez vos programmes. Les enseignants auront probablement des dizaines de programmes à noter, de sorte qu'il est difficile de repérer des similitudes entre eux. L'objectif est de prouver que vous êtes plus intelligent que l'enseignant et de ne jamais, jamais céder.