Bien communiquer avec ses enseignants d'informatique à l'université.

Camille Coti (Univ Paris13), Laure Gonnord (Univ Lyon1),
David Monniaux (CNRS), Charlotte Truchet (Univ Nantes)

23 juillet 2015

Table des matières

1 Au sujet de ce document

Ce document a été rédigé à 8 mains en juillet 2015. Il se veut une liste (non exhaustive) de conseils et de bonnes pratiques pour les étudiants d'informatique des IUT, des universités, des écoles d'ingénieurs... À lire et à relire avant toute communication avec vos enseignants.

Ce document est disponible en HTML à l'adresse `http://laure.gonnord.org/pro/conseils.html` et en PDF à l'adresse `http://laure.gonnord.org/pro/conseils.pdf`

2 Les mails en général

Lorsque vous envoyez un mail à un enseignant, il est utile de se souvenir que :

— les mails n'arrivent pas toujours (il existe dans certaines configurations de logiciel de mail/infrastructure de mail des possibilités d'avoir un accusé de réception, n'en abusez pas trop).
— l'enseignant préfère savoir à qui il/elle a affaire : utilisez votre adresse de l'Université de préférence à votre adresse personnelle du type youpi@pouetpouet.com, ce d'autant plus que ces adresses "fantaisie" peuvent provoquer un tri direct de votre courrier vers le "spam". Pour la même raison, mettez vos noms et prénoms (noms en majuscules) en signature de vos mails (ainsi que votre numéro d'étudiant, cela peut servir). Si vous mentionnez quelqu'un, précisez ses noms et prénoms.
— les conventions de politesse sont moins formelles que celles des courriers, cependant, il est préférable de respecter un minimum de forme. Pensez donc à "bonjour", "cordialement", faites des phrases (Majuscules, points) et évitez le langage SMS et les raccourcis de langage. Évitez le tutoiement.
— ce n'est pas parce que le mail permet une communication directe que l'enseignant est disponible 7j/7, 24h/24, d'autant plus qu'il/elle a souvent d'autres obligations professionnelles, voire (incroyable !) il peut avoir une vie personnelle. Laissez **toujours** un délai suffisant (quelques jours) pour répondre. Soyez certains qu'un enseignant sera profondément agacé d'une demande de renseignements envoyée la veille d'un examen, alors qu'il y avait amplement le temps avant.
— obtenir une réponse d'un enseignant n'est pas un droit, d'autant plus que les mails peuvent se perdre (par exemple lorsqu'il viennent d'une adresse inconnue comme youpi@pouetpouet.com). Sans réponse, vous pouvez relancer, mais toujours poliment.
— les enseignants ont en général des tâches variées. Il convient de mettre un peu de contexte dans votre mail, et le champ "Objet" du mail est fait pour cela. Par exemple, si vous désirez poser une question du dernier TP d'architecture (UE nommée LIF6) de Lyon1 en L2, mettez "[L2] [LIF6] Questions au sujet du TP du 12/12/2012", si vous désirez vous excuser de votre absence au cours de Réseau de première année de l'IUT de Paris 13, mettez "[IUTP13] [Réseau] Justification d'absence". En aucun cas n'envoyez de mail avec un sujet vide ou peu descriptif, comme par exemple "question".
— si vous parlez à un responsable d'UE, il peut être utile de préciser que vous parlez du cours, du TD, de quel groupe, avec quel intervenant.
— la règle "un mail = un objet" s'applique très souvent. Faire un mail avec deux informations de type différents est une mauvaise idée. Par exemple, une justification d'absence + un rendu de TP = deux mails (avec, oui, vous suivez, deux objets différents !).
— l'on attend de vous que vous lisiez vos mails universitaires régulièrement. C'est le seul moyen que l'enseignant a pour vous joindre. Si votre Université met à votre disposition une adresse mail dédiée, c'est que cela permet de vous joindre nommément sans risque d'erreur sur la personne. Vos enseignants l'utiliseront. Il est donc prudent de lire cette adresse mail suffisamment souvent (ou de la re-diriger sur un adresse privée, si c'est possible). Attention, même en cas de re-direction, il reste important de répondre avec son mail de l'université (cf plus haut).

Bien souvent, la personne qui envoie un mail pense implicitement que son interlocuteur a aussitôt lu le mail, l'a aussitôt compris, et a effectué immédiatement les démarches décrites dans le mail. C'est une fausse impression :

— Pour de nombreuses raisons, un enseignant peut manquer votre mail (parce qu'il a été classé spam, parce qu'il est arrivé au milieu d'une floppée de mails sans intérêt, etc).
— S'il l'a lu, il peut ne pas le comprendre (si votre mail vient d'une adresse du type machin@tagada.tsouin.fr et ne précise pas le contexte, si la requête ou l'information ne sont pas claires, etc). S'il ne l'a pas compris, il peut encore croire qu'il s'agit d'un spam (phishing, plaisanterie, etc) et ne pas répondre.
— S'il l'a compris, il peut noter l'information mais sans nécessairement donner suite immédiatement (s'il lit votre mail un soir ou un WE, s'il est débordé ce jour là, etc).

Bref, l'auteur du mail considère souvent qu'une fois son mail envoyé, son problème est réglé. C'est tout le contraire. Si votre mail concerne un problème important, assurez le suivi. La plupart du temps, un enseignant qui reçoit un mail y répond. Tant que votre mail n'a pas reçu de réponse, considérez qu'il n'a pas été traité.

Une dernière recommandation concernant les mails : n'en abusez pas, respectez vos interlocuteurs. Le mail doit servir à faciliter la communication **utile** entre vos enseignants et vous. Il ne doit pas servir à obtenir des passe-droits, réclamer des choses impossibles (on ne consulte pas sa copie par mail, par exemple), obtenir des informations sur le sujet d'examen, etc. D'une manière générale, tout ce que vous

n'oseriez pas demander de visu ne doit pas être demandé par mail.

3 L'orthographe, la grammaire.

De manière générale, l'orthographe et la syntaxe sont très importants pour l'impression générale de la communication que vous établissez avec votre interlocuteur. Une orthographe négligée donne l'impression que vous avez écrit à la va-vite, sans prêter attention à ce que vous écrivez. Au contraire, une syntaxe correcte et l'absence (ou une quantité faible) de fautes d'orthographe donne une impression de sérieux.

Comme le disent certains étudiants, nous ne sommes effectivement pas profs de français. Mais le recruteur qui lira votre lettre de motivation ou le client à qui vous rendrez un rapport non plus, et pourtant, un texte bourré de fautes leur donnera une très mauvaise impression de vous et votre travail.

Au delà de la mauvaise impression, un texte à la syntaxe incorrecte ou avec trop de fautes peut être difficile à lire. Certaines phrases peuvent devenir incompréhensibles.

Cela s'applique à un mail envoyé à un enseignant, à un compte-rendu de TP, à une réponse à un examen, ou à tout écrit dans le cadre de vos études ou de votre activité professionnelle. Un texte mal écrit donnera une impression de négligé et de manque de sérieux.

Pensez en particulier :

— À activer le correcteur orthographique de votre éditeur de texte ou logiciel de courrier électronique (il ne signale pas toutes les fautes, mais il est impardonnable de laisser celles qu'il signale).
— Aux majuscules, et à la ponctuation en général.
— À faire des phrases courtes, cela évite les verbes mal conjugués.
— Aux é, er, ez. (cf ce guide par exemple)
— À vous faire relire.

4 Les rendus de projet/TP.

4.1 Généralités

En général, l'enseignant qui donne un projet à faire vous indique comme le rendre : par mail, via un ENT, etc. Souvent, il précise aussi sous quelle forme il attend le rendu : types de fichiers (code et rapport, par exemple), formats et extensions (le rapport doit être en pdf, par exemple), conventions de nommage des projets (par exemple, noms des deux étudiants du binôme).

Ces consignes sont destinées d'une part, à éviter que l'enseignant perde ou mélange des projets, d'autre part, à assurer qu'il puisse ouvrir les fichiers. Elles ne sont pas facultatives ni destinées à vous ennuyer. En général, les respecter vous demande peu d'efforts. Elles sont à respecter à la lettre. Il faut insister sur ce point : **à la lettre**. En effet : rendre un tp sous la bonne forme est facile. Réparer les bêtises de 20, 40, 100 étudiants est long et fastidieux. Et **énervant**. Un enseignant énervé, c'est humain, pourra tendre à plus de sévérité dans la notation.

Le plus souvent, votre enseignant a prévu des consignes strictes pour que la correction soit facilitée, mais il peut avoir oublié des consignes. Dans ce cas, le bon sens s'applique, les paragraphes suivants développent quelques idées générales.

La question de la ponctualité mérite un point précis. Les projets sont normalement prévus sur des durées assez longues. Rendre un projet en retard, sans une raisons très très très valable :

— montre votre incapacité à organiser votre travail (il est évident qu'un projet prévu pour deux mois n'est pas faisable en une semaine),
— désorganise la correction,
— viole les règles d'égalité devant l'examen, tous les étudiants devant composer dans les mêmes conditions,
— sera, en général, pénalisé dans votre note finale sous forme de quelques points en moins (la règle étant souvent énoncée en début de projet par l'enseignant, mais pas toujours).

C'est donc déconseillé.

Dans le même ordre d'idée, en cas de retard, n'envoyez pas quinze mails larmoyants à votre enseignant. Soit vous avez une excuse qui entre dans la catégorie générale des excuses admises pour les examens, auquel cas il faut la fournir avec les preuves associées (certificat d'hospitalisation, certificat médical,

actes de décès d'un parent, etc), soit vous n'en avez pas, auquel cas, il faut et il suffit de vous excuser et d'accepter la pénalité qui suivra probablement.

Plus généralement, la plupart des enseignants universitaires sont enseignants-chercheurs et ont donc bon nombre d'activités en dehors de la simple présence devant les étudiants et le corrigé des examens et projets : administration de l'université, recherche scientifique, participation à des colloques, contacts avec des entreprises, avec souvent des délais impératifs à respecter. Ils doivent donc souvent gérer un emploi du temps compliqué ; c'est pourquoi ils peuvent ne pas pouvoir gérer des questions ou problèmes hors délai.

4.2 Le mail en lui même

- Si vous devez écrire un mail de rendu, n'oubliez pas, les consignes expliquées plus haut s'appliquent encore.
- Le mail est poli, informatif, et poli, et informatif.
- Même en cas de dépôt sur un gestionnaire de version, il peut être utile d'envoyer un mail informant que vous avez "commité" la version finale de votre projet.
- Évitez les mails "oups, j'ai oublié... ". Au besoin envoyez d'abord votre mail à un camarade pour qu'il valide que vous n'avez rien oublié.
- En cas de mail de demande de changement de date limite, considérez que ce n'est pas un dû, que vos enseignants ont aussi des dates limites pour rendre les notes, et évitez les justifications fallacieuses.

4.3 Les pièces jointes, les archives

- On rappelle qu'on ne fournit **jamais** les fichiers temporaires, les résidus de compilation : .o, fichiers tildes sont à proscrire
- Même en l'absence de consigne, nommer le tp avec son nom est LA BASE. Par exemple, tp.c : NON. $\text{tp7}_{\text{lif6gonnord}}.\text{c}$: OUI
- Même en l'absence de consigne, si votre rendu contient plus de deux fichiers, faire une archive est LA BASE. Commencez par faire un répertoire correctement nommé contenant votre projet, par exemple $\text{MIF12}_{\text{PROJETGONNORD}}$, et utilisez par exemple la commande :

```
tar cvzf MIF12_PROJET_GONNORD.tgz MIF12_PROJET_GONNORD/
```

De cette manière, votre enseignant récupérera une archive correctement nommée, et après désarchivage, un répertoire correctement nommé.

- Utilisez zip ou (de préférence) tar, et évitez RAR.
- N'oubliez pas la pièce-jointe. Vérifiez avant d'envoyer le mail. Si vous l'avez oubliée, envoyez-la tout de suite. Le fait de mettre en copie du mail les autres étudiants de votre groupe leur permet de vérifier que tout est présent dans le mail reçu par l'enseignant.

4.4 Le rendu sur un serveur distant.

Certaines Universités utilisent des plate-formes dédiées à l'enseignement, du type moodle. Si votre enseignant vous demande d'utiliser ces outils, faites encore plus attention que d'habitude à bien respecter les procédures, notamment :

- pour les binômes, si A utilise son compte (login) pour rendre le projet, n'oubliez pas de bien indiquer partout (en commentaire dans le code, en haut du rapport, etc) les noms des deux membres du binôme A et B,
- il faut parfois valider pour que le projet soit rendu (c'est le cas si un mode «brouillon» existe sur la plate-forme), assurez-vous que vous avez suivi la procédure jusqu'au bout,
- attention s'il y a plusieurs groupes dans une même UE : souvent, les projets sont à rendre groupe par groupe. Si vous rendez dans un mauvais groupe (voire un mauvais cours), votre enseignant ne pourra pas récupérer votre copie et ne saura même pas que vous avez rendu.
- enfin, vérifiez régulièrement votre compte sur la plate-forme (forums et espace de rendu / notation), de façon à pouvoir réagir si votre enseignant essaie de vous contacter.

4.5 Comment faire un bon Readme ?

Lorsque le correcteur ouvre l'archive qui contient le rendu de votre travail, il doit savoir ce qu'il est censé en faire : comment le compiler, quels sont les fichiers présents... Pour cela, la convention veut que ces informations soient données dans un fichier nommé README.

Ce fichier doit indiquer, a minima :

— le nom du ou des auteurs
— quels sont les fichiers contenus (pour vérifier qu'il n'en manque pas)
— comment compiler le code
— comment exécuter le résultat

4.6 Les rendus de code.

4.6.1 Les fichiers source (.c, .ml, .pas, .java)

Nous lisons le code que vous rendez. Il doit donc être... lisible ! C'est-à-dire :

— commenté !

Par exemple :

```
/* on alloue le tableau qui va contenir les valeurs calculées */
double* resultat = (double*) malloc( np * sizeof( double ) );
```

— avec des noms de variables et de fonctions compréhensibles. Évitez de nommer vos fonctions f1, f2 et f3...
— indenté : cela aide fortement à la lecture du code et à la compréhension de sa structuration. Par exemple :

```
for( i = 0 ; i < n ; i++ ) {
  if( tab[i] < 0 ) {
    cnt++;
  }
}
```

a une structure bien plus évidente que

```
for( i = 0 ; i < n ; i++ ) {
if( tab[i] < 0 ) {
cnt++;
}}
```

— faire simple ! une fonction pour une fonctionnalité.
— en général "optimiser" ne sert à rien. Le compilateur fait pour vous un certain nombre de choses, en particulier minimiser le nombre de variables locales ne sert à rien. Usez des variables locales, cela augmente clairement la lisibilité. De même, certaines optimisations de calcul sont réalisées à la compilation, donc jouez la carte "lisibilité" plutôt que la case performance (dans un premier temps, c'est largement suffisant).

4.6.2 Votre code doit compiler

De manière générale, quand vous devez rendre du code, il faut qu'il compile ! Le correcteur ne va pas modifier votre code pour le corriger. Il vaut mieux que vous rendiez un code qui compile mais n'implémente pas tout ce qui a été demandé, plutôt qu'un code qui implémente davantage de fonctionnalités mais ne compile pas. Pour les mêmes raisons, votre code doit s'exécuter correctement.

Vérifiez-donc cela avant de le rendre : un code qui ne compile même pas donne une impression exécrable de votre travail.

4.6.3 Les résidus de compilation (.o, .mlo, binaires)

— Un rendu de tp, ou un "commit" dans un dépôt de gestionnaire de version, ne doit pas contenir des fichiers résidus de compilation. On fera donc bien attention à supprimer les .o, les .a (sauf ceux fournis sans le source).

4.6.4 Les Makefile, les librairies externes.

— Faites attention aux dépendances cachées. En particulier, il est possible que votre compilation fonctionne à cause d'un $LD_{LIBRARYPATH}$ bien configuré sur votre machine, ou encore car une librairie est présente dans votre système. Nous vous conseillons de tester sur une autre machine que la vôtre avant de rendre votre code.
— Les Makefile les plus courts sont les meilleurs. Il est important qu'ils soient lisibles, voire un peu commentés. Si l'utilisateur de votre code doit modifier le Makefile, il convient de clairement identifier ladite partie (et d'en faire mention dans le Readme).

4.6.5 Les gestionnaires de version.

Si vous savez utiliser un gestionnaire de version, n'hésitez pas à l'utiliser. Les gestionnaires de version permettent aussi de donner des noms à des versions partielles de vos projets/tp, pensez-y.

— Les gestionnaires de version (svn, hg, git) sont là pour vous simplifier la vie. Cela dit, ils sont assez susceptibles si vous ne suivez pas les consignes de base (up à chaque fois que je commence à travailler, commit (et push) souvent, et on ne commite que des codes qui compilent). Sur des gestionnaires de version évoluées, on peut travailler sur une branche différente tant qu'une nouvelle fonctionnalité n'est pas opérationnelle, ou ne compile pas...
— Apprenez sérieusement à gérer les conflits lorsque vous n'êtes pas encore dans le rush final de rendu de tp/projet.
— N'oubliez pas de ne pas versionner les .o, fichiers temporaires, etc. Chaque gestionnaire de version a son propre système d'ignore, pour svn tapez "svn ignore" dans votre moteur de recherche favori.
— Mettez des messages de commit informatifs !
— Attention aux manipulations de répertoire. Pour svn par exemple, les répertoires versionnés contiennent des fichiers/répertoires cachés. Une manipulation du type :

```
cp -R repertoireversionne  nouveaunom
```

copie aussi ces répertoires, qui sont des fichiers de configuration svn, et votre répertoire est alors corrompu. Une solution est ce ne pas oublier de supprimer ces répertoires .svn **de la destination** :

```
find nouveaunom -type d -name .svn -exec rm -rf {} \;
```

Une autre solution est d'utiliser tar avec –exclude-vcs

— faites des commits fréquents ! Dès que vous avez ajouté une fonctionnalité, corrigé un bug... Avant de passer à la suite, faites un commit. Cela vous permettra de revenir en arrière sur une version fonctionnelle plus facilement.

4.6.6 La portabilité

— Il est possible que l'ordinateur sur lequel vous travaillez ne fonctionne pas sous le même système d'exploitation que celui de l'enseignant, ou sous la même version que celui-ci (par exemple, vous avez un Windows 10, votre enseignant un Linux Ubuntu 14.04, ou encore un MacOS X). Dans la mesure du possible, vous devrez donc éviter d'utiliser des particularismes de votre système.
— Il est possible que votre programme ne fonctionne pas sur un autre ordinateur non pas parce que celui-ci est défectueux ou a un système d'exploitation stupide, mais parce que votre programme fait quelque chose de stupide qui se trouve fonctionner par chance sur votre machine.
Par exemple, si vous ouvrez plusieurs dizaines de milliers de fichiers sans jamais les refermer, il est possible, suivant votre système d'exploitation et votre configuration, que cela fonctionne chez vous mais pas sur la machine du correcteur. Il est alors parfaitement stupide et nuisible de lui expliquer d'un air condescendant que c'est parce qu'il utilise tel système d'exploitation (p.ex. Linux) tandis que chez vous, avec un système "mieux" (p.ex. Windows), cela fonctionne mieux.
— Utiliser les outils de développement avancé, **lorsque cela a du sens**. Un éditeur un peu plus intelligent que Gedit sera utile dès que vous développez plus de 10 lignes de code, et utiliser valgrind sera utile pour débuguer d'éventuels problèmes de fuite mémoire lorsque vous travaillez sur des structures de données complexes en C.

4.6.7 Le "partage de code"

— C'est du plagiat. Vos enseignants sont habitués. Ils sont en général très méchants lorsqu'ils s'en aperçoivent. D'ailleurs, il existe des outils pour ça : celui-ci parmi d'autres.
— Pour information, si vous êtes pris la main dans le pot de confiture avec du code plagié (ou tout devoir ou rapport rendu), si votre enseignant est très gentil vous vous en sortirez avec un 0, si il l'est moins c'est passible de la section disciplinaire de l'Université. Rappelons d'ailleurs qu'une fraude à un examen comptant pour un diplôme national est un délit passible de prison. Il arrive, certes très rarement, que les universités signalent les fraudes au Procureur de la République et que celui-ci agisse.
Le texte de loi
Exemple d'étudiants ayant eu des problèmes pour ces faits

5 Les rapports de TP, de projet, de stage.

5.1 Les phrases, les figures, la mise en page.

— Tout document doit être structuré et informatif. Usez des sections avec des titres informatifs, et évitez les plans bateau. (il en est de même pour les supports de présentation, d'ailleurs). Par exemple, préférez : 1) Contexte de l'étude, problématique 2) Contribution : un nouvel algorithme polynomial pour 3-SAT 3) Preuve de correction de l'algorithme 4) Implémentation : infrastructure logicielle et résultats préliminaires ; à 1) introduction 2) travail réalisé 3) résultats.
— Pas de couleurs, on ne fait pas une oeuvre d'art. On fait un document **informatif**.
— Numérotez les pages, les figures, mettez des titres aux tableaux et aux figures, et faites référence dans le texte. Les figures sont là pour expliquer, et il faut expliquer les figures, dans le texte, en y faisant référence : "Les résultats expérimentaux sont reportés Figure 2. Sur cette figure, nous observons clairement un phénomène de seuil pour une température de 80°C. Il s'agit vraisemblablement d'un phénomène de saturation dû au fait que les extra-terrestres ne survivent pas à ces températures".
— Si vous avez eu des cours expliquant comment présenter un rapport ("Expression et Communication" en DUT par exemple), c'est le moment de les mettre en application !

5.2 La relecture

— N'oubliez pas de vous relire avant de rendre votre travail ! Si vous travaillez à plusieurs, chacun doit au minimum relire les parties rédigées par les autres.

5.3 Le contenu du rapport de tp/ de projet/ de stage

Un rapport de projet peut évidemment prendre des formes assez différentes selon la nature du projet et les habitudes de la discipline. On peut tout de même identifier quelques grandes règles.

D'abord, un rapport doit informer le correcteur sur ce qu'il a besoin d'évaluer. Ainsi, votre contribution dans le projet doit être clairement identifiée, en particulier si le projet s'inscrit dans un cadre plus large (développement d'un outil sur la base d'outils existants, par exemple). Votre contribution, c'est : ce que vous avez réussi à faire et qui était demandé, et ce qu'éventuellement vous avez réussi à faire et qui n'était pas demandé (à ne pas omettre !). Par exemple, le sujet consistait à coder un algorithme précis et que vous avez en plus ajouté une interface graphique, il faut le dire dans la contribution.

En général, le rapport doit expliquer aussi la démarche suivie, et les difficultés rencontrées. Devant un même sujet, certains étudiants commencent par coder et tester, puis affinent leur approche, d'autres modélisent, font des calculs, et codent à la fin, etc. Ce sont des informations utiles pour le correcteur, dans la mesure où cela lui permet de mieux comprendre votre travail. N'hésitez pas à décrire votre démarche, en détaillant chaque étape au besoin. Si vous avez fait des recherches sur le sujet au delà de ce qui était demandé (utilisation de documentations en ligne, bibliographie, etc), dites-le.

Si votre rapport comporte du code, celui-ci doit être suffisamment commenté (dans le rapport ET dans le code) pour être lisible. Il peut être utile de donner l'architecture de votre projet en pseudo-code pour en décrire les blocs principaux.

La conclusion d'un rapport est souvent bateau. C'est pourtant le moment de montrer vos capacités d'initiative : le projet vous a déçu sur un point que vous auriez aimé améliorer, il vous a donné des idées pour d'autres applications, il vous a donné envie de développer des fonctionnalités supplémentaires ? Dites-le ! Tout est permis, du moment que cela a du sens.

D'une manière générale, travaillez bien les parties convenues (introduction, présentation du code, conclusion). Une bonne règle est la suivante : si la phrase que vous êtes en train d'écrire pourrait être réutilisée dans un autre rapport sur un autre projet, mieux vaut la supprimer.

5.4 Le compte-rendu de TP/de projet

Un compte-rendu de TP doit refléter ce que vous avez compris des manipulations. Une liste de commandes non commentée n'a pas d'intérêt ! Dites-nous ce que vous faites et pourquoi. Par exemple, dites "on se place dans le répertoire TP3 et on compile le code fourni" et non pas "cd TP3 gcc -o exemple exemple.c" Dans le premier cas, on sait que vous avez compris ce que vous faites. Dans le deuxième, vous avez machinalement recopié une série de commandes sans l'avoir forcément comprise.

Mettre les deux est encore mieux :-)

— Un compte-rendu de TP se doit d'être informatif, recopier l'énoncé est inutile, par contre, un minimum de contexte sur les objectifs énoncés et ceux réalisés est requis.

6 Les mails de réclamations

6.1 Si vous pensez qu'il y a eu une erreur

Vous avez tout à fait le droit de penser qu'il y a eu une erreur quelque part, que ça soit dans la correction de votre copie ou dans le report des notes. Cependant, attention à ne pas être trop vindicatif quand vous demandez une vérification ! Il est possible que vous ayez tort ... Plutôt que d'écrire "je pense qu'il y a eu une erreur" (ce qui accuse directement votre interlocuteur et exclut la possibilité que vous ayez tort), demandez une vérification (ce qui laisse le champ libre à toutes les possibilités).

6.2 Demander à voir votre copie

Vous pouvez demander à voir votre copie, que ça soit pour demander une vérification de la correction ou pour savoir où vous avez commis des erreurs.

Attention cependant, certains enseignants notent large, et il n'est pas forcément dans votre intérêt de demander une nouvelle correction.

Les conseils précédents s'appliquent ici aussi : attention à ne pas être trop sûr de vous ni trop vindicatifs lorsque vous demandez à voir votre copie, et n'oubliez pas que si vous avez eu une mauvaise note, ce n'est pas forcément parce que le correcteur s'est trompé, mais cela peut aussi être parce que vous avez mal répondu à plusieurs questions !

7 Check list de survie "j'ai la flemme de lire la bafouille plus haut"

— Mon mail est informatif et signé correctement, et il est poli.

En cas de rendu de TP/projet :

— Mon mail est informatif et signé correctement, et il est poli.
— J'ai lu les consignes avant le tp/projet, et là, je viens de les relire.
— Mes fichiers sont correctement nommés.
— Mon mail ou mon compte-rendu ou mon Readme informe de mon avancée dans le tp/projet.
— J'ai relu ou fait relire mes productions écrites en français / anglais.
— Évidemment, le travail que je rends est **Personnel**