La chienlit du clavier AZERTY FR pour la programmation sous Windows. Des solutions pour retrouver le sourire.

Version 1.0 – 29/10/2024

Tous ceux qui ont utilisé un clavier Azerty FR pour programmer, ou avec un tableur, savent combien il est pénible d'aller chercher les parenthèses, les accolades et les crochets sur les touches du haut. Parfois les crochets et les accolades n'y sont même pas imprimés, ce qui rend la tâche d'autant plus pénible. La disposition Azerty FR est certainement la pire disponible pour la programmation. Aurait-on cherché à rendre nos programmeurs moins efficaces 😒

Les remèdes

- Utiliser un utilitaire pour remapper les touches du clavier (peut poser des problèmes de compatibilité, nécessité de maîtriser son fonctionnement, problèmes de pérennité...).
- Apprendre à utiliser un clavier en disposition Qwertry US voir Bépo (difficile, sauf si on commence jeune).
- Utiliser un clavier programmable pour modifier la disposition des touches (les claviers programmables en Azerty sont peu nombreux sur le marché).
- Réaliser soi-même son clavier (nécessite des connaissances en électronique, un investissement en temps, et souvent un cout assez élevé voir très élevé).
- Installer un firmware custom pour rendre programmable un clavier compatible (nécessite un investissement en temps, ainsi que l'achat d'un clavier compatible, ils sont aussi relativement rares en Azerty FR).
- **Utiliser une disposition de clavier Azerty FR modifiée dans Windows**.

J'ai essayé la plupart de ces solutions. Avec un peu de recul, deux solutions me paraissent plus avantageuses, pérennes et efficaces :

- Installation d'un firmware custom sur un clavier compatible, pour le rendre programmable.

- Utilisation d'une disposition de clavier Azerty FR modifiée dans Windows.

Installation d'un firmware custom

Si la première solution est abordable pour un programmeur, il faudra quand même trouver et aquérir un clavier compatible, réussir à le flasher avec un firmware custom et apprendre les bases de la communication entre un clavier et un système d'exploitation.

Quelques explications, des claviers, simples, mais… compliqués !

La communication clavier / système est basée, pour des raisons de compatibilité historique et technique, sur un ensemble réduit de codes 8 bits, pour les claviers PS/2 comme pour les claviers USB plus récents.

Ce codage 8 bits impose 256 codes au maximum. C'est suffisant pour une seule langue simple comme l'Anglais, mais il a fallu rapidement une adaptation logicielle pour permettre une internationalisation : la disposition de clavier. On la choisit dans le système d'exploitation, elle permet de convertir les codes claviers dans des jeux de caractères compatibles avec les langues du monde entier.

NB : Unicode, avec ses codes 16 bits, peut éventuellement être utilisé dans un clavier. Mais sans grande standardisation à ce niveau, il pose des problèmes de compatibilité. Par exemple, la méthode pour envoyer ses codes 16 bits est différente selon le système d'exploitation. Windows, Mac OS et Linux ont tous les trois une méthode différente, qui impose donc une configuration spécifique du clavier si l'on désire utiliser Unicode.

Ce n'est donc pas la solution qui est retenu pour les claviers disponibles commercialement.

Le clavier transmet donc la position de ses touches enfoncées et relâchées, et non pas un jeu de caractères. C'est ensuite le système d'exploitation qui va réaliser une correspondance entre les codes émis par le clavier (les ScanCodes), et les codes caractères affichés dans Windows (les Virtual Codes). Par exemple en Azerty FR, la touche "A" émet le ScanCode 0x10 (parfois nommé "KC_Q"), et Windows attribue le Virtual Code "VK_KEY_A". Sur un clavier Qwerty US, la même touche, marquée "Q", envoi le même ScanCode 0x10, mais Windows lui attribue le Virtual Code "VK_KEY_Q".

Apparemment simple n'est-ce pas ?

Les claviers transmettent (à quelques exceptions près) les mêmes codes pour les mêmes positions de touches, quel que soit la langue du clavier, quel que soit l'ordre et la disposition de ses touches (Azerty, Qwerty, format ISO, format ANSI...). Ce sont donc principalement les symboles imprimés sur les touches qui changent, en fonction des langues.

Au niveau du système, il existe une disposition de clavier pour chaque langue qui permet de traduire les codes claviers, les ScanCodes, en Virtual Codes. Ces ScanCodes, selon la disposition choisie dans le système, produiront donc des caractères différents.

Là où les choses peuvent se corser lorsqu'on s'intéresse à la personnalisation d'un clavier, c'est que les ScanCodes sont parfois définis sur la base de la disposition des touches du clavier Qwerty US. Dans le firmware QMK par exemple, qui est un firmware custom permettant de rendre un clavier programmable, les ScanCodes portent les noms du jeu de caractères US. Confusions assurée, bon courage pour les débutants 😡

On peut ajouter à cela la relative complexité du fonctionnement des touches modificateurs, Caps Lock, Shift, Alt, AtlGR, en effet certaines touches peuvent produire deux, trois, voire quatre caractères différents ou plus en fonction des modificateurs choisis. Des conflits entre les raccourcis clavier des applications et ceux du système peuvent rapidement apparaître sans précaution particulière. Effectuer des modifications au niveau du clavier lui-même pour changer les dispositions des touches peut donc devenir un casse-tête, même si cela reste possible.

Bref, vous l'aurez compris, il faut une certaine habitude pour trouver son chemin dans cette complexité, et à moins de s'investir un minimum, il n'est pas forcément évident de s'y retrouver à moins d'être passionné par la personnalisation des claviers.

Utilisation d'une disposition de clavier Azerty FR modifiée dans Windows

Nous en venons à la deuxième solution, qui consiste à créer une disposition de clavier modifiée pour la langue FR sous Windows. Il s'agit d'un fichier driver DLL qui s'installe automatiquement, et offre une nouvelle disposition de clavier disponible dans la barre de langues Windows.

Pour l'utilisateur final, c'est nettement plus simple, il suffit de cliquer sur le package d'installation de la disposition clavier, puis de le choisir ensuite dans la barre de langues. Il restera à éventuellement coller des stickers sur les touches modifiées, ou solution plus luxueuse, s'offrir un jeu de de touches en impression custom, si le clavier utilise des interrupteurs standards, en pratique ceux du format historique Cherry MX.

A noter, si les modifications sont mineures, il est tout à fait possible de ne rien modifier au niveau du clavier, en mémorisant simplement le nouvel emplacement des touches.

Une disposition clavier Azerty FR modifiée pour les programmeurs :

J'ai réalisé une disposition de clavier FR modifiée, à destination de nos pauvres programmeurs français, forcés d'utiliser des claviers Azerty FR, à moins d'avoir été nourris depuis le plus jeune âge à la sauce Qwerty US.

Pour rester simple et sans apprentissage, cette disposition modifiée reprend la disposition classique des lettres sur le clavier Azerty FR, en ajoutant la possibilité de changer la position de quelques touches, les plus utilisées en programmation, par une touche bascule (Modificateur Kana). Principalement : ( ) { } [ ] et quelques autres touches.

Il n'est donc pas nécessaire de réapprendre tout le clavier, les lettres restent dans l'ordre Azerty 😊

Pour faire très simple et pour éviter des conflits avec les autres modificateurs, et J'ai opté pour l'ajout d'une touche modificateur "KANA". La touche "KANA" est principalement utilisée en Japonais pour basculer entre les syllabaires **hiragana** et **katakana.** Mais rien n'empêche de l'utiliser pour d'autres langues.

J'ai utilisé la touche "Scroll Lock" qui devient donc la touche modificateur "KANA". Scroll Lock n'étant que très rarement utilisée de nos jours. Lorsque le mode "KANA" est actif, les touches ( ) { } [ ] sont remappées vers la touche entrée, comme sur un clavier US, facilement accessibles. Sinon elles sont à leur emplacement d'origine, sous les touches de fonction F1 à F12.

Si besoin la touche "Scroll Lock" demeure accessible en appuyant sur "Control" + "Shift" + "Print Screen" 🙃

NB : Pour ceux qui se demandent comment il est possible d'ajouter une double fonction à cette touche "Print Screen" normalement immuable, c'est rendu possible par une personnalisation NLS ("National Language Support", un des termes obscurs utilisé dans la personnalisation bas niveau des dispositions de clavier Windows). Les fonctions NLS servent généralement sur des claviers d'Extrême-Orient en conjonction avec un éditeur IME (Éditeur de Méthode d'Entrée) pour la saisie de caractères idéographiques complexes 😉 Mais rien n'empêche de les utiliser dans des langues occidentales, comme ici pour permettre la double fonction de la touche "Print Screen", qui devient alors une "Scroll Lock" si Control et Shift sont pressés en même temps !

Pour conclure, pour avoir les touches ( ) { } [ ] à un emplacement plus sympathique avec cette disposition, il faut simplement passer en mode KANA en appuyant sur la touche "Scroll Lock". Le mode KANA reste enclenché jusqu'à un nouvel appui.

En mode Kana, les touches situées à gauche de la touche entrée deviennent :

- touche ù -> (
- touche * ->)
- touche ^ -> [
- touche $ ->]
- la touche 6 devient un accès direct à la barre verticale -> |

Les autres touches restent inchangées, y compris en majuscules.

Toujours en mode Mode Kana, mais avec un appui sur la touche shift, on obtient les accolades et on récupère notamment les caractères "$" et ^, ainsi que d'autres caractères normalement disponibles avec le modificateur AltGR :

- touche ù -> {
- touche * -> }
- touche ^ -> ^ (direct, sans "dead character ")
- touche $ -> $
- touche 2 -> ~ (direct, sans "dead character ")
- touche 3 -> #
- touche 6 -> €

- touche 8 -> \ (anti-slash)
- touche 9 -> © (copyright)
- touche 0 -> @

Le "^", accent circonflexe, reste "^" mais devient direct, c'est-à-dire sans la fonction "dead character". C'est volontaire, parce qu'en programmation le "^" est utilisé principalement pour la fonction puissance, par exemple 56^2 = 56 au carré.

Idem pour le tilde, qui devient un tilde direct.

NB : Dans le jargon des dispositions de claviers, la fonction "dead character" aplliquée sur une touche signifie que le symbole de cette touche attend la saisie d'une seconde touche (parfois plus), pour définir entièrement le caractère et pour s'afficher. Par exemple avec "^" dans ce mode, il faut saisir ensuite une voyelle, par exemple "e", pour obtenir le caractère "ê".

Pour ceux qui se seraient posé la question, AltGR veut dire : Alternate Graphics 😋

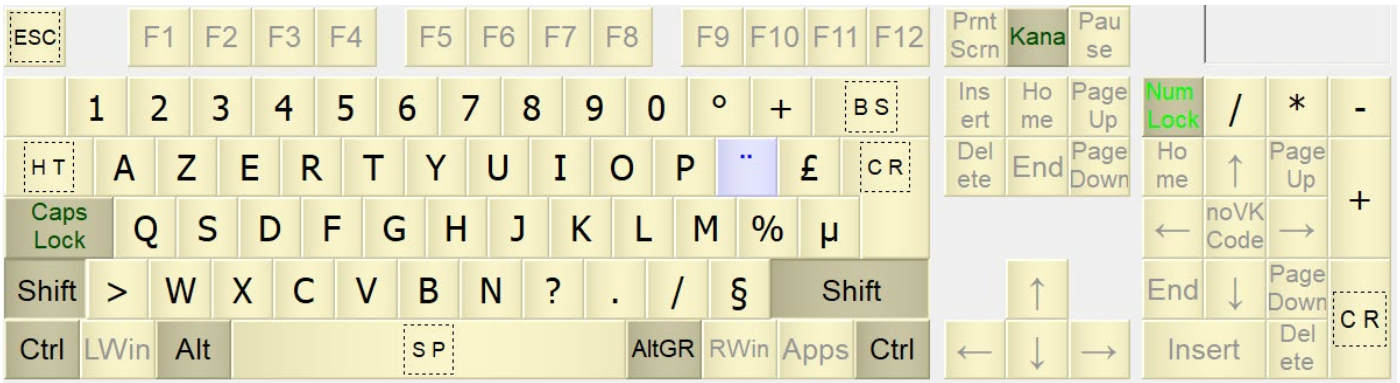
Quant à la touche "Scroll Lock", elle servait souvent dans des traitements de texte ou tableurs, pour verrouiller le défilement. Excel l'utilise toujours, un reste de préhistoire, tout comme certaines de ses fenêtres qui sont devenus incompatibles avec l'édition de longues formules pourtant encouragées par les fonctions "Lamba" et "Let" 😂.

Lorsque Scroll Lock est actif, (arrêt défilement en Français), les touches de déplacement, haut, bas, gauche et droite, déplacent la feuille au lieu de sélectionner la cellule active. Il est de plus en plus rarement utilisé.

Voici la disposition des caractères sur cette disposition Azerty FR modifiée, en fonction des modificateurs utilisés :

En mode de base sans modificateurs :

En mode shift :



En mode AltGR :

En mode Kana :

En mode Shift + Kana :