Chapitre 2

Kernel vs Nostub

Ce chapitre va nous permettre, brièvement je l'espère, de parler des deux modes de programmation qu'il existe pour nos calculatrices. Dit comme cela, vous devez probablement vous demander ce que je veux dire, mais vous comprendrez bientôt ; avant d'entrer dans les explications, je devine que vous ne vous sentirez peut-être pas très concerné par ce qui va suivre... Certes, pour des programmes que vous écrivez pour vous-même, que ce soit au cours de votre apprentissage, ou plus tard, pour votre usage personnel, le mode de programmation n'a pas une grande importance ; mais, si un jour vous êtes amené à diffuser votre, son type peut avoir une influence sur ses utilisateurs.

Au cours de ce chapitre, nous allons étudier les différences majeures entre la programmation en mode "nostub", et la programmation en mode "kernel", ainsi que ce qui peut avoir une importance pour l'utilisateur du programme.

I:\ Définitions

Tout d'abord, il serait utile de savoir ce que signifient ces deux termes.

Un programme "nostub" est un programme qui ne nécessite aucun "kernel" (noyau, en français) pour fonctionner. Cela signifie, plus clairement, que, pour utiliser ce type de programmes, il n'est pas utile d'avoir un programme tel que DoorsOS, UniversalOS, TeOS, ou encore PreOS, installé sur sa calculatrice.
Le mode nostub s'est répandu à peu près en même temps que la possibilité de coder en langage C pour nos TIs.

Un programme de type "kernel", parfois aussi dit de type "Doors", du nom du kernel qui a fixé le standard le plus utilisé, est exactement le contraire : il a besoin d'un kernel installé pour fonctionner. Notez qu'il est généralement bon d'utiliser un kernel récent, et à jour, afin de profiter de ce qui se fait de mieux.
Ce mode est, historiquement parlant, le premier à avoir été utilisé, puisqu'il remonte au temps des TI-92 simples ; notez qu'il a, naturellement, vécu des améliorations depuis.

II:\ "Kernel vs Nostub" : Que choisir ?

Chaque mode, quoi qu'en disent certains, présente des avantages, et des inconvénients. Je vais ici tenter de vous décrire les plus importants, afin que vous puissiez, par la suite, faire votre choix entre ces deux modes, selon vos propres goûts, mais aussi (et surtout !) selon ce dont vous aurez besoin pour votre programme.

Mode Kernel :

Avantages	Inconvénients
● Permet une utilisation simple de librairies dynamiques (équivalent des Dll sous Windows) déjà existantes (telles Ziplib, Graphlib, Genlib, ...), ou que vous pourriez créer par vous-même. ● Le Kernel propose de nombreuses fonctionnalités destinées à vous faciliter la vie, ainsi qu'un système d'anticrash parfois fort utile. (Une fois le kernel installé, l'anticrash l'est aussi, pour tous les programmes que vous exécutez sur la machine ; pas uniquement le votre !)	● Nécessite un programme (le Kernel) installé avant que vous ne puissiez lancer le votre. ● L'utilisation de librairies dynamiques fait perdre de la RAM lors de l'exécution du programme (parfois en quantité non négligeable) si toutes les fonctions de celle-ci ne sont pas utilisées. Cependant, notez qu'il est tout à fait possible de programmer en mode Kernel sans utiliser de librairies dynamiques ! Naturellement, la mémoire RAM est récupérée une fois l'exécution du programme terminée.

Mode Nostub :

Avantages	Inconvénients
● Ne nécessite pas de Kernel installé (Fonctionne même, normalement, sur une TI "vierge" de tout autre programme). ● En théorie, si le programme n'a pas besoin des fonctionnalités proposées par les Kernels (qu'il lui faudrait ré-implémenter !), il pourra être plus petit que s'il avait été développé en mode Kernel (car les programmes en mode Kernel sont dotés d'un en-tête de taille variable, qui peut monter à une bonne cinquantaine d'octets, et jamais descendre en dessous de environ 20-30 octets) Cela dit, en pratique, c'est loin de toujours être le cas, en particulier pour les programmes de taille assez importante.	● Ne permet pas, en ASM, la création et l'utilisation de librairies dynamiques (du moins, pas de façon aussi simple qu'en mode Kernel !) ; cela est actuellement permis en C, mais pas encore en ASM. ● En cas de modifications majeures (par Texas Instrument) dans les futures versions d'AMS, certaines fonctions d'un programme Nostub peuvent se révéler inaccessibles, et alors entraîner des incompatibilités entre la calculatrice et le programme. Il faudra alors que l'auteur du programme corrige son code et redistribue la nouvelle version du programme (Sachant que la plupart des programmeurs sur TI sont des étudiants, qui stoppent le développement sur ces machines une fois leurs études finies, ce n'est que rarement effectué !). Ce n'est pas le cas en mode Kernel, pour les fonctions des librairies dynamiques : l'utilisateur du programme n'aura qu'à utiliser un Kernel à jour pour que le programme fonctionne de nouveau.

Dans ce tutorial, nous travaillerons en mode Nostub. Non que je n'apprécie pas le mode Kernel, mais le mode Nostub est actuellement le plus "à la mode". Je me dois donc presque dans l'obligation de vous former à ce qui est le plus utilisé...
Cela dit, il est fort probable que, dans quelques temps,nous étudions pendant quelques chapitres le mode Kernel, ceci non seulement à cause de son importance historique, mais pour certaines des fonctionnalités qu'il propose. A ce moment là, nous le signalerons explicitement.
Bien que n'étudiant pas tout de suite le mode Kernel, je tiens à préciser, pour ceux qui liraient ce tutorial sans trop savoir quel Kernel installer sur leur machine (s'ils souhaitent en installer un, bien entendu), que le meilleur Kernel est actuellement PreOS, disponible sur www.timetoteam.fr.st. C'est le seul qui soit à jour : DoorsOS est totalement dépassé, TeOS l'est encore plus, de même que PlusShell, et UniversalOS n'est plus à jour. De plus, PreOS propose nettement plus de fonctionnalité que DoorsOS ou UniversalOS ! (Notons que PreOS permet naturellement d'exécuter les programmes conçus à l'origine pour DoorsOS ou UniversalOS).

Si vous faites un tour sur les forums de la communauté, vous aurez sans doute l'occasion de croiser des "fanatiques" de l'un, ou de l'autre, mode. Ne leur prêtez pas particulièrement attention ; de toute façon, si l'un vous donne dix arguments en faveur du mode nostub, l'autre vous en donnera vingt en faveur du mode kernel, et vice versa... C'est un débat sans fin, et, malheureusement, récurrent...
Comme je l'ai déjà dit, chaque mode présente ses avantages, et inconvénients ; c'est à vous, et à vous seul, de faire votre choix entre les deux, programme par programme, en pesant le pour, et le contre, de chacun.

Je n'ai pas l'intention de rendre ce chapitre plus long ; l'idée y est déjà, même si nous pourrions débattre sans fin.
Je commencerai le prochain chapitre par une brève réflexion sur la forme à donner à ce tutorial, puis nous verrons notre premier programme, ne faisant absolument rien, si ce n'est rendre le contrôle à la calculatrice une fois terminé, ce qui est une indispensable base !