Les caractères accentués et l'utilitaire Txt2bas.exe de Fabrice F.

Par André C.

Etat de la question

Txt2bas.exe est un utilitaire incontournable pour tout Oricien qui veut développer confortablement un programme Basic. En effet les possibilités d'édition incorporées au Basic de l'Oric sont plus que rudimentaires. Avec txt2bas.exe, il est possible de développer un programme dans un vrai éditeur de texte (éviter Word et consorts qui foutent des flags de formatage partout). Il vaut mieux opter pour un des très nombreux éditeurs en mode Ascii, type Bloc-note, Notepad, Notetab, etc. Je vous recommande Notepad++ <https://notepad-plus-plus.org/>, qui a été spécialement développé pour les programmeurs.

Il suffit de mouliner le fichier texte obtenu avec text2bas.exe pour obtenir un fichier tap directement utilisable. Seule ombre au tableau: les caractères accentués de Sedoric qui ne sont pas conformes au code Ascii "standard" et cela génère des problèmes.

Différentes versions de txt2bas

Dans mes archives, j'ai retrouvé 6 versions différentes de cet utilitaire. En effet Fabrice a amélioré son programme de 1995 à 2013. Au fur et à mesure qu'elles me tombaient sous la main, j'ai (mal) numéroté ces différentes versions (qui se nommaient toutes txt2bas.exe):

txt2bas1.exe (taille 44544o, daté 15/05/1998)
txt2bas2.exe (taille 46289o, daté 03/08/1995)
txt2bas3.exe (taille 116530o, daté 11/09/2002)
txt2bas3α.exe (taille 115990o, daté 19/06/2002)
txt2bas3β.exe (taille 115990o, daté 02/08/2002)
txt2bas4.exe (29245o du 16/10/2013, 32-bits)

Ces différentes versions utilisent 2 types de syntaxe:
1) Pour txt2bas1.exe et txt2bas2.exe:
txt2bas txtfile <Oric-BASIC-file>
2) Pour les 4 autres:
txt2bas [-v0|-v1|-v2] txtfile <Oric-BASIC-file>
avec: -v0 : Oric-1 keywords
-v1 : Atmos keywords
-v2 : Basic Evolution keywords

Le problème des caractères accentués

En pratique et sans entrer dans le détail, ce qui serait trop fastidieux, toutes ces versions se comportent de la même manière (j'ai vérifié) vis-à-vis des caractères accentués. Ces derniers ont été introduits par Fabrice Broche et sont mis en action par la commande ACCENT SET. Fabrice Broche ne disposait que des codes Ascii de 32 à 127, déjà utilisés pour les 96 caractères présents dans les Roms de l'Oric-1 et de l'Atmos. En effet, les codes <32 sont réservés aux codes de contrôle et attributs vidéos, tandis que les codes >127 sont réservés aux tokens Basic. Il fallait donc réutiliser les codes Ascii de quelques caractères peu utilisés, ce qui revient à redéfinir leur dessin. Ces caractères accentués sont au nombre de 6, soit:
1) à (a accent grave) redéfini à partir de @ (arobase) de code 64 (#40)
2) ç (c cédille) redéfini à partir de \ (barre oblique inverse) de code Ascii 92 (#5C)
3) é (e accent aigu) redéfini à partir de { (accolade gauche) de code Ascii 123 (#7B)
4) ù (u accent grave) redéfini à partir de | (barre verticale) de code Ascii 124 (#7C)
5) è (e accent grave) redéfini à partir de } (accolade droite) de code Ascii 125 (#7D)
6) ê (e accent circonflexe) redéfini à partir du petit damier de code Ascii 126 (#7E)

```
000 1616 1624 0000 0000 062C 0501 0000 4805 ▯▯$....▯,▯..H▯
010 6E00 9D20 506F 7572 2061 766F 6972 2022 n. Pour avoir "
020 E761 2064 6F69 7420 64E9 6AE0 20EA 7472 ça doit déjà êtr
030 6520 6CE0 206F F920 696C 2066 6175 7422 e là où il faut"
040 2064 616E 7320 6C65 2066 6963 6869 6572  dans le fichier
050 2074 6170 0091 0578 009D 2069 6C20 6661  tap.'▯x. il fa
060 7574 2074 6170 6572 2022 5C61 2064 6F69 ut taper "\a doi
070 7420 647B 6A40 207E 7472 6520 6C40 206F t d{j@ ~tre l@ o
080 7C20 696C 2066 6175 7422 2064 616E 7320 | il faut" dans
090 6C65 2073 6F75 7263 6520 7478 7400 DE05 le source txt.Þ▯
0A0 8200 BA20 22E7 6120 646F 6974 2064 E96A ..º "ça doit déj
0B0 E020 EA74 7265 206C E020 6CF9 2069 6C20 à être là où il
0C0 6661 7574 223A 2720 4176 6563 2063 6172 faut":' Avec car
0D0 6163 74E8 7265 7320 6163 6365 6E74 75E9 actères accentué
0E0 7320 7374 616E 6461 7264 002A 068C 00BA s standard.*▯..º
0F0 2022 5C61 2064 6F69 7420 647B 6A40 207E  "\a doit d{j@ ~
100 7472 6520 6C40 206F 7C20 696C 2066 6175 tre l@ o| il fau
110 7422 3A27 2041 7665 6320 6361 7261 6374 t":' Avec caract
120 E872 6573 2061 6363 656E 7475 E973 2053 ères accentués S
130 E964 6F72 6963 0000 00                   édoric...
```

Codes Ascii "Standard"

Codes Ascii "Sedoric"

Phrase d'essai: "ça doit déjà être là où il faut"

Figure 1

```
0                    MEMORY DUMP
0500: 00 48 05 6E 00 9D 20 50 6F 75 72 20 61 76 6F 69  .H.n.. Pour avoi
0510: 72 20 22 E7 61 20 64 6F 69 74 20 64 E9 6A E0 20  r "ça doit dij
0520: EA 74 72 65 20 6C E0 20 6F F9 20 69 6C 20 66 61  être là où il fa
0530: 75 74 22 20 64 61 6E 73 20 6C 65 20 66 69 63 68  ut" dans le fich
0540: 69 65 72 20 74 61 70 00 91 05 78 00 9D 20 69 6C  ier tap...x.. il
0550: 20 66 61 75 74 20 74 61 70 65 72 20 22 5C 61 20   faut taper "\a
0560: 64 6F 69 74 20 64 7B 6A 40 20 7E 74 72 65 20 6C  doit d{j@ ~tre l
0570: 40 20 6F 7C 20 69 6C 20 66 61 75 74 22 20 64 61  @ o| il faut" da
0580: 6E 73 20 6C 65 20 73 6F 75 72 63 65 20 74 78 74  ns le source txt
0590: 00 DE 05 82 00 BA 20 22 E7 61 20 64 6F 69 74 20  .Þ....º "ça doit
05A0: 64 E9 6A E0 20 EA 74 72 65 20 6C E0 20 6F F9 20  dij être là où
05B0: 69 6C 20 66 61 75 74 22 3A 27 20 41 76 65 63 20  il faut":' Avec
05C0: 63 61 72 61 63 74 E8 72 65 73 20 61 63 63 65 6E  caractères accen
05D0: 74 75 E9 73 20 73 74 61 6E 64 61 72 64 00 2A 06  tués standard.*.
05E0: 8C 00 BA 20 22 5C 61 20 64 6F 69 74 20 64 7B 6A  Œ.º "\a doit d{j
05F0: 40 20 7E 74 72 65 20 6C 40 20 6F 7C 20 69 6C 20  @ ~tre l@ o| il
0600: 66 61 75 74 22 3A 27 20 41 76 65 63 20 63 61 72  faut":' Avec car
0610: 61 63 74 E8 72 65 73 20 61 63 63 65 6E 74 75 E9  actères accentué
0620: 73 20 53 E9 64 6F 72 69 63 00 00 00 FF 00 00 00  s Sédoric...ÿ...
```

Figure 2

Nature du problème

Lorsque l'on tape un de ces 6 caractères accentués dans un traitement de texte sur un ordinateur moderne, ce caractère est codé selon un code Ascii étendu. Il en existe plusieurs et en Europe occidentale, on utilise généralement la "Page code 850". Dans ce code Ascii étendu, ces 6 caractères sont tous codés avec une valeur >127, soit:

1) #E0 (224) pour à au lieu de #40 (64)
2) #E7 (231) pour ç au lieu de #5C (92)
3) #E9 (233) pour é au lieu de #7B (123)
4) #F9 (249) pour ù au lieu de #7C (124)
5) #E8 (232) pour è au lieu de #7D (125)
6) #EA (234) pour ê au lieu de #7E (126)

Lorsque txt2bas rencontre un code >127, il le transmet tel quel. L'examen, avec un éditeur hexadécimal, du fichier tap produit permet de retrouver les valeurs #E0, #E7, #E9, #F9, #E8 et #EA à la place des caractères à, ç, é, ù, è et ê (figure 1, page précédente).

Evidemment, par la suite, la commande LIST convertit le programme en chaîne de caractères pour affichage à l'écran et l'on voit alors fleurir des trucs bizarres (figure 3, ci-dessous).

Le fonctionnement du programme est également affecté au niveau de l'affichage des chaînes de caractères lorsqu'elles contiennent des codes "standards". Quant aux REMarques, les deux commandes REM et " ' " se comportent strictement de la même manière (sans impact sur l'exécution). Mais ces remarques elles-mêmes sont souvent illisibles. Il n'en reste pas moins que les chaînes affichées dans le listing et pendant l'exécution de programme sont souillées, voire incompréhensibles. Ce n'est pas tolérable!

```
LIST
 110 REM Pour avoir "DEEKa doit dLENjL
N STR$tre lLN oRETURN WITHOUT GOSUB il
 faut" dans le fichier tap
 120 REM il faut taper "ça doit déjà ê
tre là où il faut" dans le source txt
 130 PRINT "DEEKa doit dLENjLN STR$tre
 lLN oRETURN WITHOUT GOSUB il faut":'
Avec caractLOGres accentuLENs standard
 140 PRINT "ça doit déjà être là où il
 faut":' Avec caractLOGres accentuLENs
 SLENdoric
```

Figure 3

Solution

Si l'on tient malgré tout à utiliser des caractères accentués (c'est quand même plus convivial), il y a une solution et une seule: avant de "mouliner" le fichier texte avec txt2bas, il faut utiliser la fonction "chercher/remplacer" du traitement de texte et convertir tous les :

à (a accent grave) en @ (arobase)
ç (c cédille) en \ (barre oblique inverse)
é (e accent aigu) en { (accolade gauche)
ù (u accent grave) en | (barre verticale)
è (e accent grave) en } (accolade droite)
ê (e accent circonflexe) en petit damier (caractère ~ (tilde) des PC)

Ni txt2bas, ni l'Oric ne s'apercevront du truc, car les caractères @, \, {, |,] et ~ sont parfaitement licites (code Ascii >32 et <128).

Petite illustration du phénomène

Soit le programme listé (ci-dessous). La figure 1 (page précédente) montre ce que donne l'examen du fichier tap avec un éditeur hexadécimal. On peut voir que dans la chaîne "ça doit déjà être là où il faut", les caractères accentués sont codés #E7, #E9, #E0, #EA, #E0 et #F9, qui sont bien les codes "standards". Dans la chaîne "\a doit d{j@ ~tre l@ o| il faut", on trouve les cotes #5C, #7B, #40, #7E, #40 et #7C, qui sont bien les codes "Sedoric".

```
RUN
ça doit dij@ jtre l@ oy il faut
ça doit déjà être là où il faut

Ready
```

Figure 4

```
  110 REM Pour avoir "ça doit déjà être là où il faut" dans le fichier tap
  120 REM il faut taper "\a doit d{j@ ~tre l@ o| il faut" dans le source txt
  130 PRINT "ça doit déjà être là où il faut":' Avec caractères accentués standard
  140 PRINT "\a doit d{j@ ~tre l@ o| il faut":' Avec caractères accentués Sédoric
```

```
1                          MEMORY DUMP
BBD0: 10 07 67 61 20 64 6F 69 74 20 64 69 6A 60 20 6A   ..ga doit dij` j
BBE0: 74 72 65 20 6C 60 20 6F 79 20 69 6C 20 66 61 75   tre l` oy il fau
BBF0: 74 20 20 20 20 20 20 20 10 07 5C 61 20 64 6F 69   t       ..\a doi_
BC00: 74 20 64 7B 6A 40 20 7E 74 72 65 20 6C 40 20 6F   t d{j@ ~tre l@ o
BC10: 7C 20 69 6C 20 66 61 75 74 20 20 20 20 20 20 20   | il faut
BC20: 10 07 20 20 20 20 20 20 20 20 20 20 20 20 20 20   ..                _
BC30: 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
BC40: 20 20 20 20 20 20 20 20 10 07 52 65 61 64 79 20   Figure 5  ..Ready
```

Après chargement dans la Ram de l'Atmos, le débogueur d'Euphoric permet de retrouver tous ces codes dans le programme en Ram (figure 2, page précédente, montage de plusieurs zones de dump).

Les figures 3 et 4 (page précédente) montrent ce que donnent les commande LIST et RUN. Bon, voilà nos caractères affichés: une ligne avec le codage "standard" et une ligne avec le codage "Sedoric". Il reste à vérifier, avec le débogueur d'Euphoric, les codes présents dans l'écran en Ram (figure 5, ci-dessus).

Il est réconfortant de vérifier la constance des codes (valeurs hexadécimales) aux différents niveaux (dans le fichier texte, dans le fichier tap et dans le programme en Ram après CLOAD). Mais l'Oric ne sait pas gérer les codes "standards" dans les chaînes de caractères. Ce qui est affiché par la commande LIST est une horreur, mais vous avez sans doute déjà remarqué que l'exécution du programme bogue aussi, car les codes "standards" sont remplacés par "©giyhj" (respectivement pour #E0, #E7, #E9, #F9, #E8 et #EA) (figure 4).

Petite remarque: L'examen attentif des 3 premières figures montre que j'ai fait une faute de frappe dans mon programme Basic. Et en effet, à la ligne 140 du listing, on peut voir que j'ai tapé "Sédoric" au lieu de "Sedoric". C'est l'illustration typique de ce qui peut arriver à tout moment.

Autre mystère

Nous comprenons maintenant le pourquoi de l'affichage saugrenu des LN, DEEK, LEN, LOG et STR$. Ayant rencontré les codes "standards" de à, ç, é, è et ê, soit #E0 (token de LN), #E7 (token de DEEK), #E9 (token de LEN), #E8 (token de LOG) et #EA (token de STR$), LIST incorpore le nom de ces commandes dans les chaînes de caractères qu'il envoie à l'écran. C'est son boulot normal. Mais pourquoi ce "RETURN WITHOUT GOSUB" dans le cas du code "standard" du caractère "ù"? Ce code vaut #F9 (249) et ne correspond à aucun token. Le dernier token Basic est MID$ de valeur #F7 (247).

```
LIST
 110 PRINT"#F6->A"                                  Figure 6
 110 PRINT"#F6->A"
 120 PRINT"#F7->A"
 130 PRINT"#F8->A"
 140 PRINT"#F9->A"
 150 PRINT"#FA->A"
 160 PRINT"#FB->A"
 170 PRINT"#FC->A"
 180 PRINT"#FD->A"
 200 PRINT"#FE->A"
 210 PRINT"#FF->A"
```

```
LIST
 110 PRINT"#F6->MID$"                               Figure 7
 120 PRINT"#F7->NEXT WITHOUT FOR"
 130 PRINT"#F8->SYNTAX"
 140 PRINT"#F9->RETURN WITHOUT GOSUB"
 150 PRINT"#FA->OUT OF DATA"
 160 PRINT"#FB->ILLEGAL QUANTITY"
 170 PRINT"#FC->OVERFLOW"
 180 PRINT"#FD->OUT OF MEMORY"
 200 PRINT"#FE->UNDEF'D STATEMENT"
 210 PRINT"#FF->BAD SUBSCRIPT"
```

Amusons nous un peu et essayons de voir s'il est possible de faire apparaître d'autres messages d'erreur. La figure 6 (ci-contre) montre le listing que j'ai testé. Puis avec le débogueur d'Euphoric, j'ai introduit quelques bogues: J'ai remplacé les "A" par les codes #F7 (le token de MID$), #F8, etc… jusqu'à #FF. La figure 7 (ci-conte, en bas) montre ce que devient le listing.

En résumé:
#F6 est remplacé par "MID$", comme on s'y attendait
#F7 est remplacé par "NEXT WITHOUT FOR"
#F8 est remplacé par "SYNTAX"
#F9 est remplacé par "RETURN WITHOUT GOSUB"
#FA est remplacé par "OUT OF DATA"
#FB est remplacé par "ILLEGAL QUANTITY"
#FC est remplacé par "OVERFLOW"
#FD est remplacé par "OUT OF MEMORY"
#FE est remplacé par "UNDEF'D STATEMENT"
#FF est remplacé par "BAD SUBSCRIPT"

Dans la Rom de l'Atmos, les messages d'erreur sont stockés de #C2A8 à #C3A5. Le premier message est bien "NEXT WITHOUT FOR". L'ordre des messages qui suivent est bien le même que l'ordre ci-dessus (mais après "BAD SUBSCRIPT", il manque les 10 derniers messages de la liste).

Je suppose que dans la routine de la commande LIST, il n'a pas été prévu que des octets du programme puissent être >#F6. Les développeurs de la Rom n'ont pas imaginé que cela puisse se produire un jour… avec l'arrivée de ACCENT SET !

Conclusion

Si vous voulez développer des programmes Oric comportant des caractères accentués sur un ordinateur moderne, vous devez veiller à bien remplacer les caractères "standards" à, ç, é, ù, è et ê par les caractères "Sedoric" @, \, {, |, } et ~ (ou petit damier) avec l'outil rechercher/remplacer.