Programmer le bot à l'aide d'un script

Le langage 'script' est disponible pour programmer le bot; pour ce faire,
il faut écrire et placer un fichier SCRIPT.TXT tel que celui ci-dessous
dans le répertoire du bot.

```
// Voici un script simple qui dit bonjour aux arrivants en salle

proc event (session_key, userid$, sex$, has_photo, age, is_away,
                admin, cam_on, is_bot, toc_capab, signature$, suffix$, profile_life,
                action, is_myself, line$)
{
   if (action == 128)   // l'action 128 indique une arrivée en salle
   {
      // la commande print permet d'afficher une ligne sur le chat
      print ("Bienvenue en salle " + userid$ + " :p");
   }
}
```

---

notes:
1 - si vous modifiez le script, appuyez sur la touche F5 pour le recharger.
2 - vous pouvez placer plusieurs fichiers se terminant par "SCRIPT.TXT"
dans le répertoire du bot (par ex: SCRAB-SCRIPT.TXT, ROBOT-SCRIPT.TXT),
il seront alors exécutés par ordre alphabétique.

Sur le script d'exemple, on voit une procédure "event" qui est exécutée
chaque fois qu'un évènement se passe en salle.

Cette procédure "event" reçoit les informations suivantes concernant
le chatteur concerné par l'évènement :

session_key: numéro attribué lors de l'entrée en salle d'un chatteur
et qui accompagne tous ses évènements jusqu'à sa sortie de salle;
le session_key est notamment utilisé pour désigner le chatteur à kicker.
userid$: nom du chatteur
sex$: genre du chatteur ("M", "F" ou "C")
has_photo : indique s'il a une photo dans le profil (0=non, 1=oui)
age : âge du chatteur
is_away : indique s'il est en tasse (0=non, 1=oui)
admin : indique s'il a pris son toc actuellement (0=pas de toc, 1=brun, 2=or)
cam_on : indique s'il a sa cam allumée (0=non, 1=oui, 2=oui et cam vérifiée par un anim)
is_bot : indique si c'est un bot (0=non, 1=oui)
toc_capab : indique s'il est capable de prendre son toc (0=pas de toc, 1=brun, 2=or)
signature$: signature du chatteur, exemple "1234567890ABCDEF"
suffix$: suffixe de la signature (après les deux points), exemple : "FAB2"
profile_life : nombre de jours écoulés depuis que le profil a été créé
action : type d'évènement, voir la liste ci-dessous
is_myself : indique si le chatteur est ce bot (donc moi-même) (0=non, 1=oui)
line$: contient la ligne de texte écrite par le chatteur, voir ci-dessous

actions possibles

Voici les évènements qui existent, avec leur numéro d'action :

action	évènement
0	le chatteur écrit une ligne de texte normale
1	le chatteur écrit une ligne de texte avec la bulle
2	le chatteur va ou reviens de tasse
16	le chatteur prend ou dépose le toc
127	le chatteur est déjà en salle avant l'arrivée en salle de ce bot
128	un nouveau chatteur entre en salle
129	le chatteur quitte la salle normalement / ou est téléporté dans une autre salle
130	le chatteur est déconnecté
200	le chatteur fait un PV au bot
204	le PV du chatteur a été bloqué avec le téléphone barré (line$ contient le pseudo du bloqueur)
240	un animateur donne un toc provisoire (line$ contient la session_key en 10 chiffres suivi du userid du receveur)
241	idem que 240 mais pour enlever le toc
256	le chatteur est kické hors de la salle
300	un chatteur a cliqué sur une zone de texte - voir chr$(3,a,b,c) ci-dessous
400	un chatteur a écrit un message dans le groupe (avec numéro de panneau de message (9 chiffres), numéro du message (9 chiffres), et sujet du message dans line$)
512	le chatteur parle au micro
550	le chatteur allume ou éteint sa webcam
601-609	le chatteur a cliqué sur un bouton du menu du chat (voir commandes pour créer un bouton)
912	le chatteur a cliqué sur un actor 3D (line$ contient le session_key de l'actor en 10 chiffres)
915	le chatteur a cliqué sur un objet 3D cliquable (line$ contient le numéro de l'objet 3D en 1 à 6 chiffres)

ligne de texte line$

La ligne de texte line$ contient des caractères imprimables (codes ascii 32 à 255)
mais elle contient aussi des séquences spéciales de 4 ou 8 caractères qui commencent
par un caractère entre chr$(0) et chr$(9), notamment :

chr$(0,n%256,n/256,0) : emoticon N
chr$(1,rouge,vert,bleu) : couleur (rouge vert et bleu sont des valeurs entre 0 et 255)
chr$(2,font,style,0) : changement de police ou style (voir chapitre macros)
chr$(3,a,b,c) : marqueur de zone texte - chr$(3,0,0,0) indique fin de zone
chr$(4,a,b,c) : (non utilisé pour l'instant)
chr$(5,a,b,c) : (non utilisé pour l'instant)
chr$(6,a,b,c,d,e,f,g) : image jpeg/gif/bmp/png
chr$(7,a,b,c,d,e,f,g) : son wav
chr$(8,a,b,c,d,e,f,g) : (non utilisé pour l'instant)
chr$(9,a,b,c,d,e,f,g) : (non utilisé pour l'instant)

Par exemple, si on écrit "Hello" sur le chat, userid$ et line$ peuvent contenir les caractères suivants :

userid$=[0,11,1,0,6,16,156,56,39,33,252,127,65,0,4,0,0]
line$=[0,50,0,0,0,11,1,0,6,16,156,56,39,33,252,127,65,0,4,0,0,32,58,32,1,12,94,254,2,2,0,0,72,101,108,108,111]

Voici le détail de line$:

```
0 50 0 0                : emoticon 50  ":M"
0 11 1 0                : emoticon 267 (11 + 1*256) "$nr"
6 16 156 56 39 33 252 127 : image (bannière du chatteur)
65                      : caractère A majuscule
0 4 0 0                 : emoticon 4 ";)"
32                      : un espace
58                      : caractère ':' entre le pseudo et la ligne de texte
32                      : un espace
1 12 94 254             : un code couleur (rouge=12,vert=94,bleu=254)
2 2 0 0                 : un code font (2=Arial)
72 101 108 108 111      : "Hello" en ascii
```

De façon générale, quand action==0, line$ a toujours la structure suivante :

- un emoticon (ou une image si le propriétaire de salle a dessiné ses propres icônes)
- le contenu de userid$, donc le pseudo du chatteur, qui peut être composé de plein de séquences textes, émoticons ou images,
- la séquence : espace, caractère ':', espace
- puis la ligne de texte tapée, qui peut inclure n'importe quelle séquence, y compris des changements de font ou de couleur.

Le script de test suivant peut vous permettre de faire des essais pour mieux comprendre le contenu de line$:

```
proc event (session_key, userid$, sex$, has_photo, age, is_away,
                admin, cam_on, is_bot, toc_capab, signature$, suffix$, profile_life,
                action, is_myself, line$)
{
   var i, s$, u$;
   if (!is_myself)
   {
      for (i=1; i<=len(line$); i++)
      {
         if (i > 1)
            s$ = s$ + ",";
         s$ = s$ + str$(asc(line$,i));
      }
      for (i=1; i<=len(userid$); i++)
      {
         if (i > 1)
            u$ = u$ + ",";
         u$ = u$ + str$(asc(userid$,i));
      }
      print ("action=" + str$(action) + " userid$=[" + u$ + "] line$=[" + s$ + "]");
   }
}
```

Elements du langage script

Données

Le langage script manipule deux types de données :

1) les nombres entiers :

```
9  -62  136  0xFFD2
```

Les valeurs doivent être comprises entre -2147483647 et +2147483647
Le nombre 0xFFD2 est en format hexadécimal (base 16 au lieu de la base 10 habituelle).

2) les chaines de caractères :

```
""   "Bonjour"  "Oui:p"
```

la première chaine de caractères ci-dessus est une chaine vide.
la longueur maximale d'une chaine est de 8192 caractères.

Variables

Une variable est une case de la mémoire du PC à laquelle on donne un nom;
cette case mémoire contient une valeur qui peut changer au fur et à mesure
que le programme se déroule.

Il existe 2 types de variables :

- celles qui contiennent un nombre entier (exemple: compteur, i, joueur)

- celle qui contiennent une chaine de caractère (exemple: fruit$, user$)
  (ces dernières se terminent toujours par un signe dollar '$')

Avant d'utiliser une variable, il faut la 'déclarer', c'est-à-dire réserver
de la place mémoire pour son contenu, à l'aide de la commande 'var'.

exemple:

```
var compteur;    // va contenir un compteur
var user$, t$;   // deux variables chaines de caractère
var fruit$[3];   // un tableau de 3 chaines : fruit$[1], fruit$[2] et fruit$[3]
```

Les variables peuvent être déclarées,
- soit tout en haut du script,
- soit au début d'une fonction ou d'une procédure.

Si elles sont déclarées tout en haut du script, elles gardent leur valeur
entre deux évènements; autrement, elles sont effacées à chaque fois.

Par défaut, les variables numériques ont la valeur zéro, et les variables chaines
de caractère contiennent une chaine vide.

Procédures et Fonctions

==========

Une procédure est une commande à laquelle on peut donner des paramètres,
par exemple 'print' auquel on donne une chaine de caractères "Bonjour !"
ce qui donne :

```
print ("Bonjour !");
```

Une fonction ressemble à une procédure, mais en plus elle calcule un résultat,
par exemple 'len' qui calcule la longueur d'une chaine de caractères :

```
longueur = len ("abc");
```

Les procédures/fonctions standards suivantes existent :

commandes du bot

```
print ("Bonjour !");             // le bot envoie une ligne de texte sur le chat
bubble ("pense que oui :p");     // le bot envoie une pensée (icone bulle)
away ();                         // le bot part en tasse ou revient
toc ();                          // le bot prend ou dépose son toc
kick (session_key, "va dodo");   // le bot kicke le chatteur ayant le numéro de session indiqué
```

commandes création bouton de menu

Pour créer un bouton dans le menu qui génère une action 601 à 609
quand on clique dessus :

```
create_button (session_key, 601, "{lock01}");  // avec icone
create_button (session_key, 602, ":)");        // avec smiley
create_button (session_key, 603, "T");         // avec texte
```

L'action 601 à 609 va contenir dans line$ le session_key du chatteur qui a cliqué
en 10 chiffres, suivi du texte tapé sur la ligne du chat.

pour supprimer le bouton :

```
delete_button (session_key, 601);
```

Commande Teleportation

Pour téléporter un chatteur dans une autre salle (le bot doit avoir un toc) :

```
teleport (session_key, x, y, z, dir);           // pour changer de coordonnées 3D simplement
teleport (session_key, x, y, z, dir, port);     // pas besoin d'IP si l'autre serveur est sur le même routeur ou même PC
teleport (session_key, x, y, z, dir, port, ip$);
```

calendrier

```
d$ = now$ ();          // fonction qui calcule la date du jour
                       // exemple: now$() == "2007/01/17 12:23:12"

d  = weekday (d$);     // renvoie le jour de la semaine (1=lundi, 7=dimanche) correspondant à la date
```

```
                                // exemple: weekday("2007/01/17 12:23:12") == 2

d$ = add_date$ (d$,n);  // renvoie la date vieillie de n secondes
                                // exemple: add_date$("2007/01/17 12:23:12", 3600) == "2007/01/17
13:23:12"

n = nb_days_since_1901 (d$);  // renvoie le nombre de jours entre le 1/1/1901 et la date fournie,
                                    // ce qui peut servir à calculer le nombre de jours entre deux
dates:
                                    // nb_days_since_1901 (date2$) - nb_days_since_1901
(date1$)

numérique
---------
n = random (a,b);       // renvoie un nombre aléatoire entre a et b
                                // exemple: random(1,50) == 12

n = abs (n);            // renvoie la valeur absolue de n
                                // exemple: abs(-2) == 2

n = sin (angle, M);     // calcule le sinus(angle), multiplié par M

n = cos (angle, M);     // calcule le cosinus(angle), multiplié par M

angle = atan2 (Y, X);   // calcule l'angle correspondant à l'arc-tangeante de (Y / X)

n = sqrt (n, M);        // calcule la racine carrée de n, multipliée par M


chaîne de caractère
-------------------
n  = len(s$);           // renvoie la longueur de la chaine s$
                                // exemple: len("abc") == 3

s$ = dup$(s$,N);        // renvoie la chaine s$ dupliquée N fois (avec N >= 0)
                                // exemple: dup$("ab",3) == "ababab"

s$ = chr$(N1,N2,N3,..); // renvoie une chaine constituée des caractères ascii N1,N2,..
concaténés
                                // exemple: chr$(65,66,67) == "ABC"

c  = asc(s$[,N]);       // renvoie le numéro ascii du Nième caractère de la chaine s$ (N vaut 1 si
omis)
                                // exemple: asc("A") == 65
                                // exemple: asc("ABC",2) == 66

s$ = str$(N);           // renvoie une chaine représentant le nombre N
                                // exemple: str$(23) == "23"

n  = val(s$);           // renvoie le nombre représenté par la chaine s$ (l'inverse de str$)
                                // exemple: val("23") == 23

s$ = left$(s$,N);       // renvoie les N caractères de gauche de s$
                                // exemple: left$("ABCD",2) == "AB"

s$ = right$(s$,N);      // renvoie les N caractères de droite de s$
                                // exemple: right$("ABCD",2) == "CD"

s$ = mid$(s$,a,N);      // renvoie N caractères de s$ commençant au 'a'ième
```

```
                                    // exemple: mid$("ABCDEF",2,3) == "BCD"

i  = pos(s1$,s2$);      // renvoie la position de la chaine s2$ dans s1$, ou 0 si pas trouvé
                                    // exemple: pos("ABCDEF","CD") == 3
                                    // exemple: pos("ABCDEF","X") == 0

entrée/sortie
-------------
n = count_lines ("quizz.txt");     // renvoie le nombre de lignes du fichier présent dans le
répertoire bot
                                                    // exemple: count_lines ("quizz.txt") == 120

l$ = read_line$ ("quizz.txt", i);  // renvoie la ième ligne du fichier
                                                    // exemple: read_line$ ("quizz.txt", 3) == "Question 3 :
comment s'appelle .."

i$ = image$("a.jpg");              // renvoie l'image ou le son d'un fichier présent dans le répertoire
du bot
i$ = sound$("a.wav");               // exemple: print ("une vache : " + image$("images/vache.jpg"));

s = selected_user_session_key();   // renvoie le session_key du chatteur sélectionné dans la
liste de droite
                                                    // de l'écran du bot, ou zéro si aucun chatteur n'est
sélectionné.


database
--------
store (tiroir$, valeur$);   // stocke la valeur$ dans le tiroir$ de la database chat.db
                                        // exemple: store ("JEU-DEVINE", "1 35 16 32 89 12");
                                        // la longueur de tiroir$ ne peut pas dépasser 100.

a$ = fetch$ (tiroir$);      // renvoie la valeur contenue dans le tiroir$ de la database
                                        // exemple: fetch$ ("JEU-DEVINE") == "1 35 16 32 89 12"

timer
-----
set_timer (n);     // appeler la procédure timer() dans n secondes (ou annuler si n==0)



En plus des procédures et fonctions déjà existantes décrites ci-dessus,
on peut en écrire des nouvelles; voici trois exemples :

// une fonction qui calcule le double de n
func double (n)
{
   var resultat;
   resultat = n * 2;
   return resultat;
}

// une fonction qui renvoie une chaine de N tirets
func tiret$ (n)
{
   return dup$("-", n);
}

// une procédure qui imprime N tirets
```

```
proc print_tiret (n)
{
   print (tiret$(n));
}
```

Opérateurs

Les opérateurs permettent d'effectuer des calculs, par exemple:

```
2 + 3 * 4
```

Chaque opérateur a une priorité, par exemple * a une priorité supérieure à +, ce qui veut dire que le calcul ci-dessus est équivalent à :

```
2 + (3 * 4)
```

et non pas à :

```
(2 + 3) * 4
```

Si ce dernier calcul est désiré, l'usage de parenthèses est requis.

Voici les opérateurs disponibles, par ordre de priorité décroissante :

opérateurs numériques

!	NON logique
*	multiplié par
/	divisé par
%	reste de la division
&	bitwise and
|	bitwise or
+	ajouter
-	soustraire
<=	plus petit que ou égal à
>=	plus grand que ou égal à
==	égal
!=	pas égal à
<	plus petit que
>	plus grand que
&&	ET logique
||	OU logique

opérateurs chaines

+	concaténer 2 chaines
<=	plus petit que ou égal à (ordre lexicographique = dictionnaire)
>=	plus grand que ou égal à
==	égal
!=	pas égal à
<	plus petit que
>	plus grand que

exemple:

```
if ("ab" < "abc")
   print ("ab vient avant abc dans le dictionnaire");
```

Instructions
============

l'assignation

L'instruction d'assignation permet d'effectuer un calcul (à droite)
et de stocker le résultat dans la variable (à gauche).

```
result = a * b + c;          // calcule '(a * b) + c' et stocke dans la variable 'result'
line$ = "Hello" + userid$;   // concatène "Hello" et userid$ et stocke dans line$
fruit$[2] = "pomme";         // stocke la chaine "pomme" dans la case 2 du tableau fruit$
ligne$ = "i contient la valeur " + str$(i);
longueur = len(s$) + 1;
```

L'assignation avec incrémentation permet d'ajouter ou de soustraire
une valeur à la variable de gauche.

```
i += 72;             // ajoute 72 à i
j -= (b + c);
```

L'incrémentation simple permet d'ajouter 1 ou de soustraire 1 à une variable:

```
i++;       // ajoute 1 à i
j--;       // soustrait 1 à i
```

l'instruction if

L'instruction if permet de n'exécuter un groupe d'instructions
que si une condition est remplie.

```
// if avec 1 seule instruction

if (i < 10)
   print ("i trop petit");   // n'exécuter le print que si i est plus petit que 10


// if avec plusieurs instructions
// note: des { } sont obligatoires si on a plusieurs instructions

if (i < 10)
{
   print ("i trop petit");
   print ("recommencez");
}


// "if" avec partie "else"
```

```
if (i < 10)     // si i plus petit que 10
{
   print ("i trop petit");
   print ("recommencez");
}
else          // sinon
{
   print ("i égal ou supérieur à 10 ! oué");
}


// "if" avec plusieurs parties en cascade

if (i < 10)               // si i plus petit que 10
{
   print ("i trop petit");
   print ("recommencez");
}
else if (i == 10)         // sinon si i égal à 10
{
   print ("i égal à 10 ! oué");
}
else                      // sinon (i plus grand que 10)
{
   print ("i plus grand que 10 ! oué");
}


// if avec expressions complexes

if (i < 10 && j == 3 && a$ == "hello")    // si i < 10 ET j == 3 ET a$ == "Hello"
   print ("oui");

if (i < 10 || j == 3 || a$ == "hello")    // si i < 10 OU j == 3 OU a$ == "Hello"
   print ("oui");


// if avec instruction vide

if (i < 10)
   ;         // ne rien faire
else
{
   print ("oui");
}
```

l'instruction FOR

L'instruction FOR permet de répéter l'exécution d'un groupe d'instructions entre { } en augmentant la valeur d'une variable à chaque exécution.

exemples:

```
// pour imprimer les nombres de 1 à 5 :

for (i=1; i<=5; i++)
{
   print (str$(i));
}

imprime:
   1
   2
   3
   4
   5


// pour imprimer les nombres de 5 à 1 :

for (i=5; i>=1; i--)
{
   print (str$(i));
}

imprime:
   5
   4
   3
   2
   1
```

l'instruction WHILE

L'instruction WHILE permet de répéter l'exécution d'un groupe
d'instructions tant qu'une certaine condition est vérifiée.

```
// tant que i <= 10,
// répéter les instructions entre { }

i = 1;
while (i <= 10)
{
   print ("Bonjour");
   i++;
}


// parcourir le tableau tab[]
// jusqu'à tomber sur une valeur 6

var tab[512];

.....

i = 0;
while (i <= 512 && tab[i] != 6)   // tant que i <= 512 et que tab[i] pas égal à 6, augmenter i
{
```

```
    i++;
  }
```

break et continue

```
  // l'instruction 'break' fait quitter la répétition immédiatement

  for (i=1; i<=6; i++)
  {
    if (i == 3)
      break;
    print (str$(i));
  }

  imprime:
    1
    2


  // l'instruction 'continue' fait passer au tour suivant

  for (i=1; i<=6; i++)
  {
    if (i == 3 || i == 5)   // si i égale 3 ou 5, passer au i suivant
      continue;
    print (str$(i));
  }

  imprime:
    1
    2
    4
    6
```

procédure timer
==========

Si vous ajoutez une procédure "timer" dans votre script,
celle-ci sera exécutée d'abord une fois lors de votre entrée en salle.

Ensuite, à l'aide de la procédure set_timer(N), vous pouvez faire
en sorte qu'elle soit exécutée à nouveau après N secondes.
Si N == 0, cette nouvelle exécution est annulée.

exemple

```
proc timer ()
{
  print ("ce message apparait toutes les 30 secondes");

  if (mid$(now$(),12,5) == "12:00")
    print ("il est 12 heures !");
```

```
   if (mid$(now$(),12,5) == "13:00")
      print ("il est 13 heures !");

   set_timer (30);   // exécuter timer() à nouveau dans 30 secondes
}
```

Affichage de l'icône CAM à côté du nom de la salle
==========

L'icône CAM s'affiche automatiquement à gauche du nom de la salle
lorsqu'un bot est présent en salle et configuré au moyen de bot.txt
pour imposer les webcams.

Alternativement, si vous avez votre propre script pour obliger les webcams,
vous pouvez forcer l'affichage de l'icône CAM par la commande suivante
à placer dans chatserv.ini :

```
   [settings]
   cams = oblig   ; cam obligatoire
```

Rendre une zone de texte cliquable
==========

En entourant une zone de texte par des marqueurs, on peut la rendre
cliquable, c'est-à-dire que si un chatteur clique sur la zone avec
la souris, une action 300 sera envoyée au bot contenant dans line$
le code marqueur.

Un marqueur commence par le code 3. Le marqueur de fin de zone est toujours chr$(3,0,0,0).

Le marqueur chr$(3,255,a,b) est réservé pour le jeu mine du bot.

exemples:

```
   print (chr$(3,0,0,1) + image$("logo.gif") + chr$(3,0,0,0));

   print (chr$(3,0,0,1) + image$("logo1.gif") +
            chr$(3,0,0,2) + image$("logo2.gif") +
            chr$(3,0,0,0));

   print ("Logo 1 : " + chr$(3,0,0,1) + image$("logo1.gif") + chr$(3,0,0,0) +
            "Logo 2 : " + chr$(3,0,0,2) + image$("logo2.gif") + chr$(3,0,0,0) +
            "Logo 3 : " + chr$(3,0,0,3) + image$("logo3.gif") + chr$(3,0,0,0));

...

   if (action == 300)
      print ("vous avez cliqué sur le logo " + str$(asc(line$,4)));
```

Erreurs
==========

Les erreurs suivantes peuvent apparaitre lors de l'exécution d'un script :

```
   1 "array index out of range"   : l'index du tableau [] est hors limites
   2 "division by zero"           : division par zéro non autorisée
   3 "string too long"            : chaine ne peut pas dépasser 8192 caractères
   4 "out of string space"        : trop de chaines, la mémoire est pleine
   5 "bad argument"               : mauvaise valeur pour fonction standard
   6 "application hangs"          : l'application a déjà exécuté 100 millions d'instructions
                                                 et semble tourner en rond
   7 "missing return in function" : fonction sans instruction "return"
   8 "expression too complex"     : trop de paramètres ou d'appels imbriqués
   9 "callstack full"             : trop de paramètres ou d'appels imbriqués
```

Les erreurs 10 à 15 dénotent un problème avec le logiciel de chat :

```
  10 "arithmetic stack empty"
  11 "unknown pcode",
  12 "outside code range"
  13 "outside var range"
  14 "ret outside range"
  15 "astack corrupt"
```

Exemples de scripts
==========

//

```
//  Exemple 1 : Horloge

proc event (session_key, userid$, sex$, has_photo, age, is_away,
                  admin, cam_on, is_bot, toc_capab, signature$, suffix$, profile_life,
                  action, is_myself, line$)
{
   var heure, phrase$, phrase2$;

   if (action == 128)   // arrivée en salle
   {
      heure = val (mid$(now$(),12,2));

      if (heure >= 12 && heure <= 13)
         phrase$ = "Bon Appétit$mi ";
      else if (heure >= 0 && heure <= 4)
         phrase$ = "Agréable Nuit(s) ";
      else if (heure >= 6 && heure <= 19)
         phrase$ = "Bonjour:) ";
      else
         phrase$ = "Bonsoir;) ";

      if (sex$ == "F")
         phrase2$ = " (f)";
      else
         phrase2$ = " (y)";


      print (phrase$ + userid$ + phrase2$
                  + ", il est " + right$(now$(),8)  + "(o)");
```

```
   }
}

////////////////////////////////////////////////////////////////////////////

//  Exemple 2 : jeu de Puissance 4

var jeu;         // 1 = jeu actif
var tab[35];     // 5 lignes de 7 colonnes
var joueur;


proc effacer_table ()
{
   var i;
   for (i=0; i<35; i++)
      tab[i] = 0;
}


proc afficher_table ()
{
   var line, col, line$, i;

   i = 0;

   for (line=0; line<5; line++)
   {
      line$ = "   ";

      for (col=0; col<7; col++)
      {
         if (tab[i] == 0)
            line$ = line$ + chr$(1,0,0,0) + ".";
         else if (tab[i] == 1)
            line$ = line$ + chr$(1,0,0,255) + "X";
         else
            line$ = line$ + chr$(1,255,0,0) + "O";
         i++;
      }

      print (chr$(2,5,4,0) + line$);  // courier new + gras
   }

   print (chr$(2,5,4,0) + chr$(1,0,0,0) + "  >" + "1234567<");
}


func score (col, line)
{
   var x, y, c, i, j, count;

   // 4 cases consecutives horizontales

   j     = 0;
   count = 0;
```

```
for (x=0; x<7; x++)
{
   c = tab[x+line*7];

   if (c == 0 || c != j)
   {
      count = 1;
      j = c;
   }
   else
   {
      count++;
      if (count == 4)
         return j;
   }
}


// 4 cases consecutives verticales

j     = 0;
count = 0;

for (y=0; y<5; y++)
{
   c = tab[col+y*7];

   if (c == 0 || c != j)
   {
      count = 1;
      j = c;
   }
   else
   {
      count++;
      if (count == 4)
         return j;
   }
}


// 4 cases consecutives obliques \

j     = tab[col+line*7];
count = 1;

for (i=1; i<=3; i++)
{
   if (col-i < 0 || line-i < 0 || tab[col-i + (line-i)*7] != j)
      break;
   count++;
}

for (i=1; i<=3; i++)
{
   if (col+i >= 7 || line+i >= 5 || tab[col+i + (line+i)*7] != j)
      break;
```

```
        count++;
    }

    if (count >= 4)
        return j;


    // 4 cases consecutives obliques /

    j     = tab[col+line*7];
    count = 1;

    for (i=1; i<=3; i++)
    {
        if (col-i < 0 || line+i >= 5 || tab[col-i + (line+i)*7] != j)
            break;
        count++;
    }

    for (i=1; i<=3; i++)
    {
        if (col+i >= 7 || line-i < 0 || tab[col+i + (line-i)*7] != j)
            break;
        count++;
    }

    if (count >= 4)
        return j;

    return 0;
}


// col de 1 a 7
// joueur 1 ou 2

func ajouter (col, joueur)
{
    var line;

    col--;

    if (col < 0 || col >= 7)
        return -1;

    if (tab[col] != 0)
        return -1;

    for (line=0; line<4; line++)
    {
        if (tab[col+line*7+7] != 0)
            break;
    }

    tab[col+line*7] = joueur;

    return score (col, line);
}
```

```
proc event (session_key, userid$, sex$, has_photo, age, is_away, admin,
                cam_on, is_bot, toc_capab, signature$, suffix$, profile_life,
                action, is_myself, line$)
{
   var c$, col, r;

   if (action == 128)
   {
      print ("Bienvenue en salle, tapez !p4 pour jouer au puissance 4.");
   }

   if (action == 0)
   {
      if (len(line$) >= 3 && right$(line$,3) == "!p4")
      {
         jeu = 1;
         afficher_table();
      }
      else if (jeu == 1)
      {
         if (len(line$) >= 1)
         {
            c$ = right$(line$, 1);
            if (c$ >= "1" && c$ <= "7")
            {
               col = val (c$);

               if (joueur == 1)
                  joueur = 2;
               else
                  joueur = 1;

               r = ajouter (col, joueur);

               if (r >= 0)
               {
                  afficher_table();

                  if (r > 0)
                  {
                     if (r == 1)
                     {
                        print (chr$(1,0,0,0) + "Bravo ! " + chr$(1,0,0,255) + "X"
                                 + chr$(1,0,0,0) + " a gagne");
                     }
                     else if (r == 2)
                     {
                        print (chr$(1,0,0,0) + "Bravo ! " + chr$(1,255,0,0) + "O"
                                 + chr$(1,0,0,0) + " a gagne");
                     }

                     effacer_table ();
                     jeu = 0;
                  }
```

```
            }
          }
        }
      }
   }
}

////////////////////////////////////////////////////////////////////////////////


// Exemple 3 : kicker les tasses après 30 secondes

var session[256];     // enregistre le session_key des chatteurs en tasse
var expire$[256];     // date et heure à laquelle on kicke le chatteur
var nb_session;       // nombre d'éléments dans les 2 tableaux


// ajoute une session_key dans les 2 tableaux

proc ajouter_dans_table (session_key)
{
   var i;

   for (i=0; i < nb_session; i++)
   {
      if (session_key == session[i])    // trouvé
         break;
   }

   if (i == nb_session)   // pas trouvé, donc agrandir table
      nb_session++;

   session[i] = session_key;
   expire$[i] = add_date$ (now$(), 30);
}


proc supprimer_de_table (session_key)
{
   var i;

   for (i=0; i < nb_session; i++)
   {
      if (session_key == session[i])   // trouvé
         break;
   }

   if (i < nb_session)   // trouvé
   {
      session[i] = session[nb_session-1];
      expire$[i] = expire$[nb_session-1];
      nb_session--;
   }
}


proc timer ()
{
```

```
   var i;

   for (i=0; i < nb_session; i++)
   {
      if (now$() > expire$[i])   // a expiré
      {
         kick (session[i], "ce n'est pas un parking ici !");
         supprimer_de_table (session[i]);
      }
   }

   set_timer (1);    // rappeler timer() toutes les secondes
}


proc event (session_key, userid$, sex$, has_photo, age, is_away,
                  admin, cam_on, is_bot, toc_capab, signature$, suffix$, profile_life,
                  action, is_myself, line$)
{

   if (!is_myself)  // pas moi-même
   {
      if (action == 2)
      {
         if (is_away)   // chatteur va en tasse
         {
            print ("Attention " + userid$ + " : tu seras kické après 30 secondes en tasse !");
            ajouter_dans_table (session_key);
         }
         else          // reviens de tasse
         {
            supprimer_de_table (session_key);
         }
      }

      if (action == 129 || action == 130 || action == 256)
      {
         supprimer_de_table (session_key);
      }
   }

}


////////////////////////////////////////////////////////////////////////////////

// Exemple 4 : robot multi-fonctions

var nombre;              // nombre a deviner (pour le jeu devine)
var jeu_score[50], jeu_pseudo$[50];

var nombre_de_lignes;    // compte le nombre de lignes reçues

var fruit$[3];           // tableau de 3 fruits


func fq$ ()    // changer la couleur et le style d'écriture
{
```

```
   return chr$(1,0,0,128)  // couleur bleue claire (voir chapitre macros)
          + chr$(2,5,4,0);   // font "Courier New", style gras
}

func red$ ()
{
   return chr$(1,255,0,0);   // couleur rouge
}

func green$ ()
{
   return chr$(1,0,255,0);
}

func blue$ ()
{
   return chr$(1,0,0,255);
}


proc demarre_devine ()
{
   print ("devine un nombre de 1 à 99");
   nombre = random (1,99);
}


// ajoute des points dans le tableau jeu_score[]

proc ajoute_points (userid$)
{
   var i, j, u$;

   // cherche le userid$ dans le tableau jeu_pseudo$[]
   // ou bien cherche une case vide dans le tableau

   for (i=0; i<50; i++)
   {
      if (jeu_pseudo$[i] == userid$ || jeu_pseudo$[i] == "")
         break;
   }


   // remplir la case et augmenter le score

   jeu_pseudo$[i] = userid$;
   jeu_score[i]++;            // ajouter 1 point au score


   // remonter le joueur dans le tableau des scores
   // s'il a dépassé des joueurs avant lui

   while (i > 0 && jeu_score[i] > jeu_score[i-1])
   {
      j = jeu_score[i];
      jeu_score[i] = jeu_score[i-1];
      jeu_score[i-1] = j;

      u$ = jeu_pseudo$[i];
```

```
      jeu_pseudo$[i] = jeu_pseudo$[i-1];
      jeu_pseudo$[i-1] = u$;

      i--;
   }
}


// imprime les 9 premiers scores du tableau
// ainsi que le userid$ même s'il est après les 9 premiers

proc affiche_scores (userid$)
{
   var i, score$;

   print (fq$() + "SCORES");
   print (fq$() + "======");

   for (i=0; i<50; i++)
   {
      if (jeu_pseudo$[i] == "")
         break;
      if (i < 9 || userid$ == jeu_pseudo$[i])
      {
         score$ = str$(jeu_score[i]);
         if (len(score$) < 4)
            score$ = dup$(" ",4-len(score$)) + score$;
         print (fq$() + str$(i+1) + ". " + score$ + "  " + jeu_pseudo$[i]);
      }
   }
   print("");
}


proc joue_devine (word$,userid$)
{
   var last, first, n;

   // enlève les blancs à la fin
   last = len(word$);
   while (last >= 1 && asc(word$,last) == 32)
      last--;

   // collecte tous les chiffres à la fin
   first = last;
   while (first >= 1 && asc(word$,first) >= 48 && asc(word$,first) <= 57)
      first--;

   if (first == last)   // pas de chiffres
      return;

   n = val (mid$(word$,first+1,last-first));

   if (n < nombre)
      print ("ha non, trop petit !");
   else if (n > nombre)
      print ("hé non, trop grand !");
   else
```

```
   {
      print ("Bravo, c'était bien " + str$(nombre));
      ajoute_points (userid$);
      affiche_scores (userid$);
      nombre = 0;
   }
}


proc print_fruits ()
{
   if (fruit$[1] == "")   // tableau pas encore rempli
   {
      fruit$[1] = "Pommes";
      fruit$[2] = "Poires";
      fruit$[3] = "Bananes";
   }
   print ("Saluuuut :d  tu aimes les " + fruit$[random(1,3)] + " ? ");
}



// nettoye la ligne line$ en remplaçant toutes les icones par des blancs,
// en convertissant tous les mots en minuscules,
// et en entourant tous les mots par des blancs.

func clean$ (line$)
{
   var last, r$, i, c;

   i = 1;
   last = len(line$);

   while (i <= last)
   {
      c = asc (line$,i);
      if (c <= 9)  // séquence spéciale
      {
         i += 4;    // passer 4 caractères
         if (c > 5)
            i += 4;
         c = 32;  // remplacer par un blanc
      }
      else   // caractère normal
      {
         if (c >= 65 && c <= 90)  // si entre A et Z
            c += 32;    // convertir en minuscules
         i++;
      }
      r$ = r$ + chr$(c);
   }

   return r$ + " ";
}
```

```
proc timer ()
{
   if (mid$(now$(),12,5) == "12:00")
      print ("il est 12 heures !");

   if (mid$(now$(),12,5) == "13:00")
      print ("il est 13 heures !");

   set_timer (55);  // rappeler toutes les 55 secondes,
}                   // donc 1 appel par minute est garanti


proc event (session_key, userid$, sex$, has_photo, age, is_away,
                  admin, cam_on, is_bot, toc_capab, signature$, suffix$, profile_life,
                  action, is_myself, line$)
{
   var word$, w$, heure, i;

   word$ = clean$(line$);   // nettoye la ligne pour pouvoir y chercher des mots

   if (action == 127)   // afficher les gens déjà présents en salle
   {
      print ("Déjà en salle : " + userid$ + "  (session_key=" + str$(session_key) + ")");
   }

   if (action == 128)   // arrivée en salle
   {
      if (is_myself)    // c'est moi-même
         toc();         // je prend le toc !

      w$ = blue$();

      heure = val (mid$(now$(),12,2));

      if (heure >= 12 && heure <= 13)
         w$ = w$ + "Bon Appétit$mi ";
      else if (heure >= 0 && heure <= 4)
         w$ = "Agréable Nuit(s) ";
      else if (heure >= 6 && heure <= 19)
         w$ = w$ + "Bonjour:) ";
      else
         w$ = w$ + "Bonsoir;) ";

      w$ = w$ + userid$;

      if (sex$ == "F")
         w$ = w$ + " (f)";
      else
         w$ = w$ + " (y)";

      w$ = w$ + ", il est " + right$(now$(),8)  + "(o)";

      print (w$);

//    print ("Votre signature est : " + signature$);

      print (blue$() + "tape " + red$() + "!devine" + blue$() + " pour le jeu devine");
```

```
}

if (action == 0)    // ligne de texte normale
{
   if (!is_myself)    // ce n'est pas moi-même
   {
      if (pos (word$, " !devine "))    // contient !devine (avec 2 blancs autour, attention !)
         demarre_devine ();

      if (nombre > 0)  // le jeu devine est actif
         joue_devine (word$, userid$);


      // kicker quelqu'un s'il dit cochon

      if (pos (word$, " cochon "))
         kick (session_key, userid$ + ":@ malpoli !!");


      // Afficher l'heure

      if (pos (word$, " heure "))
         print ("Horloge parlante : " + now$());


      // afficher le fichier bienvenue.txt, ligne par ligne

      if (pos (word$, " bien "))
      {
         for (i=1; i<=count_lines("bienvenue.txt"); i++)
            print (read_line$("bienvenue.txt", i));
      }


      // Afficher une image de porte

      if (pos (word$, " porte ") > 0)
         print ("-> " + image$("kitchen02.jpg") + "quelle porte ? :p");


      // retenir une phrase en la stockant dans la database chat.db

      if (pos (word$, " retiens ") > 0)
      {
         store ("Retiens " + signature$, line$);
         print ("phrase retenue, dis 'quoi' pour la revoir !");
      }

      // redire la phrase stockée

      if (pos (word$, " quoi ") > 0)
      {
         print ("ta phrase était : "+ fetch$("Retiens " + signature$));
      }


      nombre_de_lignes++;    // augmenter le nombre de lignes de 1
```

```
            // toutes les 200 lignes, annoncer le nombre de lignes déjà parlées en salle

            if ((nombre_de_lignes % 200) == 0)
               print ("Déjà " + str$(nombre_de_lignes) + " lignes parlées sur ce chat :p");


            // toutes les 100 lignes, demander si on aime un fruit

            if ((nombre_de_lignes % 100) == 0)
               print_fruits ();
         }
      }


      if (action == 2 && is_away)   // aller en tasse
      {
         print ("Et alors là " + userid$ + " ?  on se cache ?");
      }

      if (action == 2 && !is_away)  // revenir de tasse
      {
         print ("Ben heureusement que tu reviens " + userid$ + " $si");
      }

      if (action == 1)       // ligne de texte envoyé avec la bulle
      {
         print ("Alors on rêvasse " + userid$ + " :d");
      }
   }

   ////////////////////////////////////////////////////////////////////////////////
```