

## TXT\_HAM

Ce logiciel permet de préparer des exercices ou des informations prêts à être intégrés dans une hyperbase.

Le programme txt\_ham s'organise autour d'un éditeur. Outre les problèmes généraux (charger, sauver, ...) les deux fonctions principales sont:

**Analyse:** cette fonction contient les rubriques:

*Contrôle de la description:* contrôle de la syntaxe,

*Préparation des adresses:* les adresses physiques sont établies,

*Sauvetage et récupération du code intermédiaire:* pour contrôle.

Le système signale les erreurs. Par ailleurs, il crée un fichier d'extension MSG contenant quelques avertissements complémentaires.

**Insérer dans hyperbase:** procède à l'insertion proprement dite.

Toutes les informations à intégrer sont contenues dans des fichiers texte. Le nombre de noeuds par fichier n'est pas limité. Il est toutefois conseillé de créer un fichier différent pour chaque exercice !

Un fichier est constitué de deux parties: la déclaration des hyperbases utilisées et une suite de noeuds. Ces deux parties sont séparées par un point virgule.

La déclaration des hyperbases est une suite d'éléments de la forme: **identificateur = nom de fichier**, séparés par des virgules.

Les noeuds sont également séparés par des point-virgules.

Un noeud est de la forme:

**numéro / base**

> **suite d'attributs séparés par des virgules**

> **suite de contextes séparés par des virgules**

> **donnée**

> **suite de liens séparés par des virgules**

Par la suite quelques exemples montrent l'ensemble des possibilités. A la fin du document on trouvera une description détaillée de la syntaxe utilisée.

### Exemple d'un exercice pour ACCORDE

Code	Remarques
théorie = "THEORIE.HBA", exercice = "EXERCICE.HBA" ;	déclaration des hyperbases utilisées.
exercice_3 / exercice > - , "printname /Exercice de vocabulaire", "type /exercice", "window/23/23/10/3/37/5"	Nom du noeud / base Liste des attributs. - signifie qu'un attribut name = nom du noeud est ajouté Ce noeud est un exercice

<pre> &gt; "0 interaction type qrep",   "1 interaction mode normal",   "2 interaction feed_back oui",   "2 interaction feed_back non",   "2 interaction tolérance minu",   "2 interaction tolérance maju",   "0 experts dom français",   "0 experts did français",   "0 experts mode ppassé",   "2 interaction reprise avec",   "2 interaction reprise sans",   "2 interaction reprise 3",   "1 contrôle ordre original"  &gt; "Tapez la forme conjuguée convenable!"  &gt; question refq1 / exercice question refq2 / exercice question refq3 / exercice  aide_stat # ui_acc0 / théorie exemple # ui_exem / théorie hlp_guide # ui_guid / théorie méthode # ui_meth / théorie ; </pre>	<p>Liste des contextes</p> <p>La consigne est le contenu</p> <p>Liste des liens. Il y a trois liens sur des questions.</p> <p>Les autres liens sont sur des aides déjà existantes dans la base, ce qui est signalé par #</p> <p>; termine ce noeud</p>
<pre> ' mode normal; à taper la réponse  refq1/ exercice &gt; "type /question" &gt; &gt; "tapez" &gt; énoncé refe1 / exercice réponse refr1 / exercice aide_stat # ppassé / théorie msg # msg_f2 / ex_base "err_typ /mangeait" ; </pre>	<p>Ce qui suit ' est un commentaire !</p> <p>Nouveau noeud</p> <p>C'est une question</p> <p>Sans contexte particulier</p> <p>Consigne liée à la question</p> <p>Lien sur un énoncé, une réponse et de la théorie déjà existante.</p> <p>Référence à un noeud de type objectif. Message en cas de réponse « mangeait ».</p>
<pre> refe1 / exercice &gt; "type /donnée", &gt; &gt; ! "le cheval % de l'herbe tendre": "(manger, passé-composé)" : "" &gt; ; </pre>	<p>Nouveau noeud</p> <p>Donnée servant d'énoncé</p> <p>! indique une donnée</p>
<pre> refr1 / exercice &gt; "type /donnée", &gt; &gt; ! "a mangé" &gt; ; </pre>	<p>Noeud contenant la réponse</p>
<pre> ' réponse par qcm  refq2 / exercice &gt; "type /question" &gt; "1 interaction mode qcm 3" &gt; "choisissez la bonne réponse" &gt; énoncé refe2 / exercice réponse refr2 / exercice distracteur refr2_1 / exercice distracteur refr2_2 / exercice </pre>	<p>Nouvelle question</p> <p>Elle modifie l'interaction de l'exercice en introduisant un qcm</p> <p>Lien sur un énoncé, une réponse, trois distracteurs et un exemple déjà existant.</p>

distracteur refr2_3 / exercice exemple # ui_exem2 / théorie ;	
refe2 / exercice > "type /donnée" > > ! "le cheval % de l'eau clair": "(boire, passé composé)" : "" > ;	Enoncé de la question précédente.
refr2 / exercice > "type /donnée", > > ! "a bu" > ;	Réponse
refr2_1 / exercice > "type /donnée", > > ! "a bue" > ;	Donnée servant de distracteur (mais pouvant servir de réponse pour une autre question!)
refr2_2 / exercice > "type /donnée" > > ! "est bu" > ;	Distracteur
refr2_3 / exercice > "type /donnée" > > ! "est bue" > ;	Distracteur
' mode tableau  refq3 / exercice > "type /question" > "1 interaction mode tableau 2 2 masculin/féminin singulier/pluriel" > "tapez les éléments manquants" > énoncé vide / exercice réponse refr3 / exercice exemple # ui_exem3 / théorie ;	Question introduisant l'interaction tableau  Le nom vide pour l'énoncé est important.
vide / exercice > "type /donnée" > > ! "" > ;	En principe on ne fabrique qu'une donnée vide par hyperbase!
refr3 / exercice > "type /donnée" > > ! ["cheval" "jument" "chevaux" "juments"] >	La réponse servira à construire l'énoncé!

### Exemple d'un exercice pour LACUNE

Code	Remarques
------	-----------

<pre> théorie = "THEORIE.HBA" , exercice = "EXERCICE.HBA" ; exercice_2 / exercice &gt; - ,   "printname /Exercice lacunaire",   "type /exercice"  &gt; "0 interaction type lacune",   "0 experts dom lacune",   "0 experts did lacune",   "1 contrôle ordre original"  &gt; "Retapez ou complétez les réponses!"  &gt; question refq1 / exercice question refq2 / exercice aide_stat # ui_lacu / théorie exemple # ui_exem / théorie hlp_guide # ui_guid / théorie méthode # ui_meth / théorie ; </pre>	déclaration des hyperbases utilisées.
<pre> refq1 / exercice &gt; - ,   "type /question",   "remp /*" &gt; "0 interaction mode enigme", &gt; "enigme" &gt; énoncé refe1 / exercice ; </pre>	
<pre> refe1 / exercice &gt; "type /donnée",   "window /92/92/4/0/18/80",   "name/texte 1" &gt; &gt; ! @ "q1.txt" &gt; ; </pre>	Le contenu est dans le fichier « q1.txt »
<pre> refq2 / exercice &gt; "type /question",   "remp /*",   "aremp /aon on ont" &gt; "0 interaction mode syllabe", &gt; "Remplacer les lacunes par aon, on ou ont" &gt; énoncé refe2 / exercice exemple # ui_exem2 / théorie ; </pre>	
<pre> refe2 / exercice &gt; "name /texte 2",   "type /donnée",   "window /92/92/4/0/18/80" &gt; &gt; ! @ "q2.txt" &gt; </pre>	

### Exemple d'un exercice pour FLASH-VOC

Code	Remarques
------	-----------

théorie = "THEORIE.HBA", exercice = "EXERCICE.HBA" ;	déclaration des hyperbases utilisées.
<pre>exercice_1 / exercice &gt; - ,   "printname /Exercice de vocabulaire",   "type /exercice"  &gt; "0 interaction type flash",   "0 experts dom français",   "0 experts did flash_voc",   "2 interaction position centré",   "2 interaction position aléatoire",   "1 contrôle ordre aléatoire",   "2 interaction temps 75",   "2 contrôle paramètre 0 100"  &gt; "Retapez ou complétez les réponses!"  &gt; question refq1 / exercice   question refq2 / exercice   aide_stat # ui_voca / théorie   exemple # ui_exem / théorie   hlp_guide # ui_guid / théorie   méthode # ui_meth / théorie ;</pre>	
<pre>'à retaper de façon identique  refq1/ exercice &gt; "type /question" &gt; &gt; nil &gt; énoncé refe1 / exercice ;</pre>	
<pre>refe1 / exercice &gt; "type /donnée",   "lisibilité /20" &gt; &gt; ! "le cheval" &gt; ;</pre>	
<pre>'à taper une autre réponse suggérée par une amorce  refq2 / exercice &gt; "type /question" &gt; &gt; nil &gt; énoncé refe2 / exercice   réponse refr2 / exercice   exemple # ui_exem2 / théorie ;</pre>	
<pre>refe2 / exercice &gt; "type /donnée",   "lisibilité /20" &gt; &gt; ! "le cheval": "les" : "" &gt; ;</pre>	
<pre>refr2 / exercice &gt; "type /donnée", &gt;</pre>	

> ! "chevaux" >	
--------------------	--

## Exemple d'un exercice pour PROBLEME

Code	Remarques
théorie = "MATHBAS0.HBA" , exercice = "PROBLEME2.HBA" ;	déclaration des hyperbases utilisées.
# ui_pb2 / théorie > > > nil > x sys1 / exercice "name/exercice" ;	Ce noeud existe déjà. On veut y crocher l'exercice sys1.  En mettant x à la place du handle, le système proposera un choix !
sys1 / exercice > - , "printname /Quelques exercice plus difficiles", "type /exercice", "window/23/23/10/3/37/5"  > "0 interaction type qrep", "1 interaction mode normal", "1 interaction feed_back non", "1 experts dom calcul", "1 experts did calcul", "1 experts mode système2x2", "2 interaction reprise avec", "2 interaction reprise sans", "2 interaction reprise 3", "1 contrôle ordre original"  > "Donnez la valeur de x et y"  > question sys1q1 / exercice aide_stat # ui_sys2x2 / théorie méthode # ui_méth_qrep_écrire / théorie donCmpl cmpl_sys1 / exercice ;	L'unité ui_sys2x2 « sait » utiliser les variables qui apparaîtront par la suite
cmpl_sys1 / exercice > - , "type /donnée" > > ! nil, "dimX" ["quelconque"], "dimY" ["quelconque"], "b" [si ( sval_var "s1" "-", b1 * -1 , b1 )], "d" [si ( sval_var "s2" "-", d1 * -1 , d1 )] > ;	Donnée complémentaire  Donnée de type « formule ». La première partie (ici nil) donne le calcul à effectuer sur la réponse. La deuxième partie sur les données
sys1q1/ exercice > "type /question" > > nil > énoncé sys1e1 / exercice réponse sys1r / exercice ;	
sys1e1 / exercice > "type /donnée", > > ! " % x % % y = %\n % x % % y = %" :	Enoncé:  0.2 x + b1 y = 0.8 0.2 x + 0.3 y = v

<pre>"a" "s1" "b1" "u" "c" "s2" "d1" "v" : "a" [0.2] , "b1" [0.4, 0.5], "c" [0.2], "d1" [0.3], "u" [0.8] , "v" [0.5, 0.7], "s1" ["+"], "s2" ["+"] &gt; ;</pre>	<p>b1 vaudra 0.4 ou 0.5 v vaudra 0.5 ou 0.7</p>
<pre>sys1r / exercice &gt; "type /donnée", &gt; &gt; ! [x y] &gt;</pre>	<p>Réponse: valeur de x et de y</p>

## Exemples d'unité d'information de théorie

Code	Remarques
<pre>théorie = "MATHBAS0.HBA" ui_sol_chinoise_add / théorie &gt; -, "type /objectif", &gt;window/113/23/5/0/6/80", &gt; printname/Inspiré du Jiuzhang suanshu, 1er siècle av. J.C. &gt; (Chine)" &gt; &gt; " 2 boeufs et 3 moutons valent 13 taels; 3 boeufs et 2 moutons valent 12 taels. Combien valent respectivement un boeuf et un mouton? ~ Solution &lt;023sol&gt; " &gt; sol ui_sol_chinoise_add_sol / théorie "name/descripteur" ;</pre>	<p>déclaration de l'hyperbase</p> <p>Le terme 'Solution' sera un hyperchamp. Couleur 023 (écriture blanche sur fond bleu). Le concept associé est 'sol'.</p>
<pre>ui_sol_chinoise_add_sol / théorie &gt; -, "type /objectif", &gt;window/3/113/2/0/22/80", &gt; printname/Méthode d'addition à la chinoise" &gt; &gt; " Problème avec notre notation: 2x + 3y = 13 3x + 2y = 12 etc ... " &gt; ;</pre>	

**Note concernant les messages:** des noeuds de type objectif peuvent servir de message. L'attribut `err_typ` sur le lien indique le moment où le message est délivré. Soit en cas de réponse juste si la valeur de l'attribut est `juste_typ` ou de réponse fausse si la valeur est `err_typ`. Sinon le message est délivré lorsque la réponse est justement la valeur de cet attribut. Ces liens peuvent figurer au niveau de la question, d'un exercice ou même d'un noeud de la carte des connaissances. Il est donc possible de complètement modifier la famille des messages.

## Grammaire de description de données pour Prof'Expert

BNF	Explication
DOC ::= BASEL ; NOEUDL	Document
BASEL ::= BASE + separator ,	liste des bases séparées par ,
BASE ::= IDENT = STRING	exemple: exercice = « EX.HBA »
NOEUDL ::= NOEUD + separator ;	les noeuds sont séparés par ;
NOEUD ::= BASE_REF > STRLIST > STRLIST > CONTENU > LIENS	référence du noeud liste des attributs liste des contextes contenu liste des liens
BASE_REF ::= NUMERO / IDENT	exemple: ui_3 / théorie
NUMERO ::= # IDENT   REF   IDENT	
STRLIST ::= STRING   - * separator ,	
CONTENU ::= nil   STRING   # BASE_REF   @ STRING   ! @ STRING   ! DON	# contenu d'un noeud existant @ nom d'un fichier ! donnée
DON ::= STRING : STRINGLIST : LVALEUR   STRING : STRINGLIST : STRING   STRING   EXP , LVALEUR   EXP   [ DONL ]	problème expr2 expr formule exp list
DONL ::= DON +	liste de données
LIENS ::= LIEN *	liste de liens
LVALEUR ::= VALEUR + separator ,	liste de valeurs séparées par des virgules
LIEN ::= IDENT BASE_REF STRLIST	un lien est constitué d'un handle, de l'adresse du noeud cible et d'un attribut
STRINGLIST ::= STRING *	liste de chaînes
REF	Exemple: 123
IDENT	Exemple: toto_1
STRING	Exemple: « voici une chaîne »
LEXP ::= EXP + separator ;	liste d'expressions séparées par ;
<b>Pour mathématique</b>	
LEXP_COND ::= EXP_COND + separator ,	liste d'expressions conditionnelles séparées par ,
EXP_COND ::= EXP   si ( COND , EXP )   si ( COND , EXP , EXP )   STRING : STRINGLIST   STRING   si ( COND STRING STRINGLIST )	expression math. sans de condition si (exist « a », 3) si (sval_var « s1 » « - », -b1, b1) sans condition (expr2) sans condition (expr) expr2 avec condition
COND ::= EXP > EXP   EXP >= EXP   EXP = EXP   exist STRING   sval_var STRING STRING	la variable STRING existe la variable STRING vaut STRING
VALEUR ::= STRING = [ LEXP_COND ]   STRING [ LEXP_COND ]	a = [ 2, 3, 4 ] possibilités pour a
EXP ::= EXP + EXP   EXP - EXP -- EXP / EXP   EXP * EXP   EXP : EXP -- EXP ^ EXP -- ( EXP ) = ( EXP )   ( EXP )   - EXP   ent EXP   # HEURE   √ RACINE   [ EXPL ] ? STRING   nil   EXP_BASE	exemples: 2 + 3 ; a - 2 ; (x+1)/(x-2) ; x(x+1) ; x : 45 ; 2^x ; (3x+y) = (x+y) (3x^2 - 1) ; - (x - 1) ; ent 3.5 ; # 3 h 4 ; √ 2 (carrée) ; √ 2 (cubique)
EXP_BASE ::= LONG % LONG   LONG % REAL   LONG / LONG   LONG / LONG LONG   LONG /LONG REAL	prendre les 3% de 250: 3 % 250 fraction 2/3 prendre les deux tiers de 12: 2/3 12 prendre les deux tiers de 3.5: 2/3 3.5

LONG / - LONG   LONG   IDENT   \$ IDENT   REAL % LONG   REAL % REAL   REAL   enum ( EXP_BASE_L )   rnd ( EXP_BASE EXP_BASE )   rnd   pgdc ( EXP_BASE EXP_BASE )   ppmc ( EXP_BASE EXP_BASE )   eucl ( EXP_BASE EXP_BASE )   mod ( EXP_BASE EXP_BASE )	fraction 2/-3 23 ; x ; \$ x (littéral x)  enum (3, 4, 5, 4.5) rnd ( 10 20) pgdc( 24 36)  division euclidienne reste de la division
RACINE ::= LONG   REAL     LONG    REAL	
HEURE ::= EXP_BASE h EXP_BASE min   EXP_BASE h EXP_BASE   EXP_BASE h   EXP_BASE min EXP_BASE sec   EXP_BASE min EXP_BASE   EXP_BASE min   EXP_BASE sec	
EXP_BASE_L ::= EXP_BASE + separator ,	
Pour français	
VALEUR ::= STRING = [ LEXP]	
EXP ::= STRING   STRING : STRINGLIST	