KAI ⊆ TEX

唐

坤爱

Jeudi 15 juin 2000


慨 开 锯 恺 楷 揩 铠 蛤 汽
凯 垲 剀 铠

Mode d'emploi du format kai

1. Syllabes chinoises

La commande `\kainames` donne la liste des noms des syllabes chinoises disponibles ; soit, les 406 suivantes :

•a,ai,an,ang,ao•ba,bai,ban,bang,bao,bei,ben,beng,bi,bian,biao, bie,bin,bing,bo,bu•ca,cai,can,cang,cao,ce,cen,ceng,ci,cong, cou,cu,cuan,cui,cun,cuo•cha,chai,chan,chang,chao,che,chen, cheng,chi,chong,chou,chu,chuai,chuan,chuang,chui,chun,chuo•da, dai,dan,dang,dao,de,dei,deng,di,dia,dian,diao,die,ding,diu, dong,dou,du,duan,dui,dun,duo•e,en,er•fa,fan,fang,fei,fen,feng, fo,fou,fu•ga,gai,gan,gang,gao,ge,gei,gen,geng,gong,gou,gu,gua, guai,guan,guang,gui,gun,guo•ha,hai,han,hang,hao,he,hei,hen, heng,hng,hong,hou,hu,hua,huai,huan,huang,hui,hun,huo•ji,jia, jian,jiang,jiao,jie,jin,jing,jiong,jiu,ju,juan,jue,jun•ka,kai, kan,kang,kao,ke,ken,keng,kong,kou,ku,kua,kuai,kuan,kuang,kui, kun,kuo•la,lai,lan,lang,lao,le,lei,leng,li,lian,liang,liao, lie,lin,ling,liu,long,lou,lu,luan,lun,luo,lv,lve•m,ma,mai,man, mang,mao,me,mei,men,meng,mi,mian,miao,mie,min,ming,miu,mo,mou, mu•na,nai,nan,nang,nao,ne,nei,nen,neng,ng,ni,nian,niang,niao, nie,nin,ning,niu,nong,nou,nu,nuan,nuo,nv,nve•o,ou•pa,pai,pan, pang,pao,pei,pen,peng,pi,pian,piao,pie,pin,ping,po,pou,pu•qi, qia,qian,qiang,qiao,qie,qin,qing,qiong,qiu,qu,quan,que,qun• ran,rang,rao,re,ren,reng,ri,rong,rou,ru,ruan,rui,run,ruo•sa, sai,san,sang,sao,se,sen,seng,si,song,sou,su,suan,sui,sun,suo • sha,shai,shan,shang,shao,she,shei,shen,sheng,shi,shou,shu, shua,shuai,shuan,shuang,shui,shun,shuo•ta,tai,tan,tang,tao,te, teng,ti,tian,tiao,tie,ting,tong,tou,tu,tuan,tui,tun,tuo•wa, wai,wan,wang,wei,wen,weng,wo,wu•xi,xia,xian,xiang,xiao,xie, xin,xing,xiong,xiu,xu,xuan,xue,xun•ya,yan,yang,yao,ye,yi,yin, ying,yo,yong,you,yu,yuan,yue,yun•za,zai,zan,zang,zao,ze,zei, zen,zeng,zi,zong,zou,zu,zuan,zui,zun,zuo•zha,zhai,zhan,zhang, zhao,zhe,zhei,zhen,zheng,zhi,zhong,zhou,zhu,zhua,zhuai,zhuan, zhuang,zhui,zhun,zhuo•ponct.

2. L'environnement kai

C'est l'environnement dans lequel on saisit les "codes kai", c'est à dire les codes associés aux caractères de l'alphabet kai. Le début de l'environnement est indiqué par la commande `\kai` et la fin par la barre oblique '`/`'. Le contenu de l'environnement est une liste

indéfiniment longue de codes kai entourés par des espaces ou des retours de chariot. La saisie se présente donc sous la forme :

```
\kai
<code-kai> <code-kai> <code-kai> <code-kai>
<code-kai> ..................... <code-kai>
<code-kai> <code-kai> <code-kai> <code-kai>
/
```

Codes kai

Un code kai est constituée de trois champs contigus

`<syllabe><accent><indice>`

avec

- `<syllabe>` : l'un des noms donnés par `\kainames` ;
- `<accent>` : un nombre à un chiffre parmi `{0,1,2,3,4}` ;
- `<indice>` : un nombre entier positif.

Exemple : `bang15` est le code du caractère d'indice `5` de la syllabe `bang` munie de l'accent `1` ; `\kai bang15 /` donne 拌.

Assistance à la saisie

En raison du très grand nombre de codes kai (actuellement 7219), nous avons prévu un système d'aide en ligne basé sur l'incomplétude des codes kai. Nous décrivons le mode de fonctionnement en rajoutant des codes kai à la saisie :

`\kai bang15 /`

Il nous faut d'abord un nom de syllabe. L'astérisque : `'*'` équivaut à `\kainames` et :

`\kai bang15 * /`

affiche la liste des noms des syllabes. On choisit alors un nom, par exemple `'liao'`, et notre ligne devient :

`\kai bang15 liao /`

dont la compilation donne :

拌(liao)

(liao1) 撩
11

(liao2) 僚 嘹 寥 寮 廖 撩 潦 燎 獠 疗 缭 聊 辽 鹩
21 22 23 24 25 26 27 28 29 210 211 212 213 214

(liao3) 燎 蓼 钌 了
31 32 33 34

(liao4) 尥 撂 料 钌 镣
41 42 43 44 45

Cette présentation donne les accents de la syllabe concernée (`1`, `2`, `3` et `4` dans l'exemple) et, pour chaque accent, les caractères disponibles. On choisit un caractère, par exemple celui d'indice `10` de `liao2` et l'on complète la saisie en

```
\kai bang15 liao210 /
```

dont la compilation donne : 拌疗. On passe alors à la saisie du code kai suivant.

Remarque : Lorsque `<syllabe><accent>` est présent mais que l'on omet `<indice>`, on voit apparaître uniquement la ligne du tableau précédent concernant l'accent en question. La compilation de

```
\kai bang15 liao2 /
```

donne

拌(liao2) 僚 嘹 寥 寮 廖 撩 潦 燎 獠 疗 缭 聊 辽 鹩
1 2 3 4 5 6 7 8 9 10 11 12 13 14

où il convient d'observer que les chiffres indiqués sous les cases représentent, dans tous les cas, les paramètres qui manquent pour former le code kai du caractère choisi.

Remarque technique : Lorsque la liste des codes kai est réduite à un seul élément, la commande `\kai <code-kai> /` peut être remplacée par

```
\kaimacro{<syllabe><accent>}<indice>,
```

forme intrinsèquement plus rapide et robuste que la précédente puisque dépourvue de la phase de balayage de l'environnement dont

la complexité pourrait poser des problèmes dans vos définitions de macros TEX.

A propos du contenu de l'environnement kai

En dehors des codes kai, cet environnement peut contenir d'autres éléments ; son contenu général est représenté par :

```
\kai
    | <code-kai> | * |    cr     |    ,;:.?!~    |
    | 0123456789 | \ | \kaispace | text:<chaîne> |
/
```

- `cr` : introduit un passage à la ligne sans changement de paragraphe.

  ```
  \noindent\kai
  ang41 ang41 ang41 ang41 cong17 cr
  ang41 ang41 ang41 ang41 ang41 cong17
  /
  ```

  donne :

  盎盎盎盎驄
  盎盎盎盎盎驄

  Cette manière de coder le passage à la ligne est à préférer à la suite classique `\hfill\break` pour des raisons de compatibilité avec le modes d'écriture verticale dans lequel le même mot `'cr'` commandera le changement de colonne.

- **Ponctuation et nombres dans l'environnement kai.** Un signe de ponctuation ou un nombre peut être tapé directement à condition qu'il soit entouré par des espaces ou des retours de chariot. Ces éléments seront insérés dans une fonte romane de taille et gras correspondants à celle de la fonte kai courante. Les exemples suivants expliquent son utilisation :

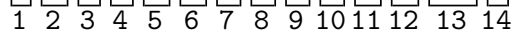
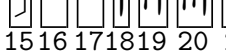
  `\kai ang41 : 2001 . cang22 \dots /` donne : 盎: 2001. 藏...
  `\kai ang41 , cang22 chui12 . /` donne : 盎, 藏炊.

Remarque : On pourrait être tenté d'insérer des ponctuations en sortant de l'environnement kai,

`\kai ang41 /, \kai cang22 chui12 /.`

donne '盎, 藏炊.' ce qui est bien proche de l'affichage précédent mais pas tout à fait identique. En effet, il faut bien comprendre que dans le premier type de saisie la ponctuation et l'espace qui le suit seront de taille correspondante à celle de la fonte kai utilisée ce qui n'est pas nécessairement le cas dans la deuxième saisie. Par exemple, en utilisant une fonte kai plus grande, les deux saisies en question donnent respectivement :

盎, 藏炊. et 盎, 藏炊.

Remarque : Le nom de syllabe '`ponct`' donne accès aux ponctuations de la fonte kai. On a : `(ponct0)` 
1 2 3 4 5 6 7 8 9 10 11 12 13 14

15 16 17 18 19 20 21 22 23 24 25 26.
Enfin, la commande `\kaipoint` produit un point circulaire : `\kai ang41 \kaipoint cang22 chui12 \kaipoint /` donne 盎。藏炊。

- **Caractères latins dans l'environnement kai**
  Une conséquence des facilités de la saisie des signes de ponctuation et nombres est que si l'on saisit une chaîne de caractères ne correspondant pas à un nom de syllabe, celle-ci sera affichée dans la fonte romane courante. Par exemple :
  `\kai\kailarge\guil Alberto \guir = a14 le41 bei412 tuo11 /`
  donne : 〝Alberto〞=阿乐贝托
  (Les guillemets ont été obtenus en posant `\def\guil{ponct016}` et `\def\guir{ponct017}`.)

  Toute chaîne de caractères ne correspondant pas à un code kai est donc affichée en caractères latins. Si vous désirez éteindre cette option aux noms de syllabe vous devez utiliser la syntaxe

‘`text :<chaîne>`’ où `<chaîne>` désigne une suite de caractères ne comportant aucun espace ou retour de chariot.

La saisie : `\kai a21 text :a a21 /` donne donc 啊a啊.

Remarque technique : Augmentons d'un facteur de cinq l'espace intercaractère. Les saisies :

`\kai ai411 , ai412 /` et `\kai ai411 ,, ai412 /`

donnent alors : 艾, 隘 et 艾 ,, 隘

On remarquera que les espaces avant et après ‘`,`’ et ‘`,,`’ ne sont pas les mêmes. Dans le premier cas, le format reconnaît qu'il s'agit d'une ponctuation, la colle au caractère précédent et l'éloigne du suivant en conséquence. Dans le second cas, la suite de caractères est gérée comme un caractère kai ordinaire, elle est donc précédée et suivie de l'espace intercaractère. On considère qu'un caractère latin est une ponctuation lorsque sa largeur est inférieur à la moitié de la taille de la fonte kai courante, sa largeur est alors redéfinie au 33% de la taille de la fonte kai courante.

Remarque. Le caractère `~` qui représente traditionnellement une espace insécable est gérée comme une ponctuation.

- **Espacements : ‘`\ `’ et `\kaispace`.**
  La commande `\setkaiintercharkern=<dim>` fixe l'écart être deux caractères kai successifs à une largeur égale à `<dim>` (plus un leger mou lui permettant la justification). Sa valeur par défaut est le `10%` (voir `\kaiintercharfactor`) de la taille de la fonte kai courante. Dans le cas présent la commande `\kai` donne

  隘隘隘隘隘隘隘隘隘隘

  `\kai{\setkaiintercharkern=0pt}` aurait donné

  隘隘隘隘隘隘隘隘隘隘

  et `\kai{\setkaiintercharkern=10truept}` donne

  隘 隘 隘 隘 隘 隘 隘 隘 隘 隘

  Remarque : Tout caractère kai (sauf le dernier dans la commande `\kai.../`) est **suivi** de l'espace intercaractère du style utilisé.

Dans l'environnement kai on dispose de deux types d'espaces, celui qui sépare deux caractères kai consécutifs (accessible par la commande `‘\ ’`), et celui qui suit une ponctuation qui vaut le 33% de la taille de la fonte kai courante (accessible par `\kaispace`), mais il n'est pas prévu d'espaces aux frontières de l'environnement. En effet, la dernière opération de l'environnement `\kai.../` consiste à détruire la dernière espace horizontale. (Il s'ensuit qu'il suffit de rajouter une espace supplémentaire : `\kai...\ /`, pour préserver la dernière espace.)

Voici différentes manières de produire des espaces frontalières :

`a\kai ai412 /b`	donne	a隘b
`a\kai \ ai412 \ /b`	donne	a隘b
`a \kai ai412 / b`	donne	a 隘 b
`a\kai \kaispace ai412 \kaispace \ /b`	donne	a 隘 b

Malgré le peu de différences que l'on observe entre les premières et dernières lignes, ces différences sont bien réelles ; pour peu que l'on augmente la taille des caractères kai, on les aperçoit mieux :

`a\kai ai412 /b`	donne	a隘b
`a\kai \ ai412 \ /b`	donne	a隘b
`a \kai ai412 / b`	donne	a 隘 b
`a\kai \kaispace ai412 \kaispace \ /b`	donne	a 隘 b

C'est à l'utilisateur de décider quel type d'espacement frontalier convient mieux à sa mise en page.

Macros à l'intérieur de l'environnement kai

L'environnement kai peut tolérer des commandes TEX dans les conditions suivantes :

- **Macros sans paramètre.** En principe sans exception ; l'exemple typique est donné par le changement de paragraphe sans alinéa :

```
\kai
ang41 cang22 chui12 keng13  lve42 hang24  ang41 lve42
ang41 cang22 chui12 keng13  lve42 hang24 keng13 lve42
ang41 cang22 chui12 keng13  lve42 chui12 keng13 lve42 .
\noindent
ang41  cang22 chui12 keng13 lve42 hang24 ang41  ang41
cang22 chui12 keng13 lve42 hang24  ang41 ang41 cang22
cang22 chui12 keng13 lve42 hang24  ang41 ang41 cang22 .
/
```

ce qui donne :

盎藏炊铿略行盎略盎藏炊铿略行铿略盎藏炊铿略炊铿略.

盎藏炊铿略行盎盎藏炊铿略行盎盎藏藏炊铿略行盎盎藏.

- **Les macros d'alias.** Nous avons prévu un moyen commode pour créer des raccourcis pour une liste de codes kai qui revient souvent. La commande de base est :

  `\kaidef#\<alias>{<liste de codes kai>}` (∗)

  où `#` est l'un des chiffres `0` ou `1`.

  Une translittération de "Alberto" est "`a le bei tuo`" et s'écrit "阿乐贝托", ce qui résulte de `\kai a14 le41 bei412 tuo11 /`. On déclare alors :

  ```
  \kaidef0\alberto{a14 le41 bei412 tuo11}
  ```

  et une saisie de la forme :

  ```
  Alberto ignore que \alberto\ est son nom chinois.
  ```

  donne :

  Alberto ignore que 阿乐贝托 est son nom chinois.

  La même macro d'alias peut être utilisée à l'intérieur d'un environnement kai :

```
\kai \alberto  ai412 ai412 ai412 ai412 ai412
\alberto  ai412 ai412 ai412 ai412 ai412
\alberto ai412 /
```

donne :

阿乐贝托隘隘隘隘隘阿乐贝托隘隘隘隘隘阿乐贝托隘

Remarque : L'expansion d'une macro `\<alias>` définie par la commande `\kaidef#` dépend de l'endroit où elle se trouve et du chiffre '`#`' qui suit `kaidef`.

– A l'extérieur d'un environnement kai, elle donne

`\kai <liste de codes kai> /` lorsque `#=0` ;

`\kai <liste de codes kai> \ /` lorsque `#=1` ;

de sorte que `#=1` est à utiliser lorsque l'on souhaite que l'espace suivant le dernier caractère kai ne soit pas supprimée. On reviendra sur ce détail plus tard.

– A l'intérieur d'un environnement kai, elle donne uniquement la liste des codes, quel que soit le chiffre `#`. Par exemple :

```
\kai ao23 \<alias> ao43 /                    (‡)
```

donnera

`\kai ao23 <liste des codes kai> ao43 /`

Il convient d'observer qu'une définition standard :

`\def\<alias>{\kai <liste de codes kai> \ /}`

aurait fait de (‡) une saisie équivalente à :

`\kai ao23 {\kai <liste des codes kai> \ /} ao43 /`

et l'espace intercaractère avant `ao43` aurait été préservée.

En fait, la vraie raison pour préférer `\kaidef#` à `\def` est en relation avec les environnments kai verticaux, où, dans la mesure où l'expansion donnera uniquement la liste des codes kai, le résultat sera affiché également en mode verical.

- **Macros avec paramètres.** Uniquement si la macro et ses paramètres sont entre accolades ; en particulier une saisie de la forme :

  `\kai{\<macro>{#1}{#2}...} ... /`

  est acceptée.

Recommandation : Nous conseillons d'éviter de mettre à l'intérieur de l'environnement kai autre chose que ce qui est prévu dans ce manuel ; en cas de doute nous conseillons de quitter l'environnement kai pour y revenir ultérieurement. Cette recommandation est en tout cas **impérative** pour les formules mathématiques. La bonne saisie pour obtenir :

盎藏炊 $\alpha \cong \beta$ 铿略行

est donnée par :

```
\kai ang41 cang22 chui12 / $\alpha\cong\beta$
\kai keng13 lve42 hang24 /
```

- **Concaténation des macros d'alias.** Le sujet abordé dans le paragraphe sur les espacements prend toute son importance lors de la concaténation des macros d'alias. En effet, dans ce cas l'espace intercaractère doit être restituée entre deux alias successifs. Deux solutions se présentent naturellement.

  A l'aide de `kaidef1` : suite aux définitions suivantes :

```
\kaidef1\alberto{ao23 ao43 lan37}
\kaidef1\al{ao23}                          (*)
\kaidef1\berto{ao43 lan37}
```

  `\vrule\alberto\ et \al\berto.` donne (en multipliant par 10 l'espace intercaractère pour mieux apprécier le résultat) :

  |敖 奥 览 et 敖 奥 览 .

  où l'on remarquera l'espace supplémentaire avant 'et' et '.'.

  C'est un résultat logique dans la mesure où l'espace en fin d'alias est préservée précisément pour garantir des concaténations cor-

rectes (`\al` et `\berto`). La recette pour supprimer ces espaces finales passe par l'utilisation de la primitive de TEX `\unskip` qui supprime toute espace la précédant.

La ligne `\vrule\alberto\ et \al\berto.` modifiée en :

`\vrule\alberto\unskip\ et \al\berto\unskip.`

donne alors un résultat juste : |敖　奥　览 et 敖　奥　览.

Moralité : En utilisant `kaidef1`*, on peut concaténer sans soucis des alias de codes kai, mais il faudra intervenir manuellement par un* `\unskip` *pour supprimer une espace finale.*

Lorsque l'alias est utilisé uniquement en mode horizontal, une définition standard : `\def\<alias>{\kai ... \ /}` *donnera les mêmes résultats que* `kaidef1`*.*

A l'aide de `kaidef0` : on remplace `#=1` par `#=0` dans les définitions (∗) ; `\vrule\alberto\ et \al\berto.` donne alors :

|敖　奥　览 et 敖奥　览.

et l'on se retrouve avec le problème inverse où il faut reconstituer l'espace intercaractère séparant `\al` de `\berto`.

La valeur exacte de cette espace est obtenue par `\kai \ \ /`, et le format kai propose `\def\kaisp{\kai \ \ /}` de sorte que si l'on modifie `\vrule\alberto\ et \al\berto.` en

`\vrule\alberto\ et \al\kaisp\berto.`

on retrouve le bon résultat : |敖　奥　览 et 敖　奥　览.

Moralité : En utilisant `kaidef0`*, aucune espace n'apparaît en fin d'alias, par contre il faudra intervenir manuellement avec un* `\kaisp` *pour rajouter l'espace intercaractère entre deux alias consécutifs.*

Lorsque l'alias est utilisé uniquement en mode horizontal et à l'extérieur des environnements `\kai.../`*, une définition standard :* `\def\<alias>{\kai ... /}` *donnera le même résultat que* `kaidef0`*.*

Conclusion : Nous ne proposons aucune solution universelle à ce type problème. C'est à l'utilisateur d'évaluer, pour un alias donné, la solution la plus économique. Nous pensons, pour notre part, qu'un alias destiné essentiellement à être concaténé, doit comporter une espace à sa fin.

- **Paramètres globaux.** Les valeurs de `10%` et `33%` relatives à l'espace intercaractère et après une ponctuation sont fixées dans le format kai par les définitions :
  - `\def\kaiintercharkernfactor{0.1}`
  - `\def\kaispacefactor{0.3333}`

  Pour que ces définitions soient prises en compte par un environnement kai, elles doivent être extérieures à l'environnement (contrairement au mode d'utilisation de `\setkaiintercharkern` par exemple). Lorsque placées en tête de votre travail, leur effet portera sur tous les environnements kai. Par contre, une saisie de la forme :

```
{\def\kaispacefactor{2}
      a\kai\kaispace ai412 \kaispace\ /b}
```

  donne : a 隘 b, et la modification ne porte sur aucun autre environnement kai en raison des accolades `{...}`.

Mode horizontal rigide

Nous n'avons pas prévu d'alignement vertical par défaut des caractères kai dans l'environnement kai horizontal puisque dans les paragraphes où coexistent, chinois, latin et mathématiques, l'esthétique globale nous a parue meilleure.

La commande

`\kairigid{<facteur>}`

où `<facteur>` est un nombre réel qui représente la largeur (relative à la taille du style courant) de la cellule contenant un caractère kai,

permet de résoudre ce problème.
Par exemple `\kai ... / ... \kai ... /` donne :

肋肋肋: 肋肋肋肋肋.
肋肋; 肋肋肋肋肋肋肋. 肋肋肋肋肋肋肋肋了了了了了了1991了了了了了了了了 $x^n + y^n \neq z^n$ pour $n > 2$ et $x, y, z \neq 0$ 了了了了了了了了了了了了了了了了了肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋肋

où l'on remarquera le manque d'alignement vertical, tandis que `\kai{\kairigid{1.1}} ... /` donne :

肋肋肋: 肋肋肋肋肋.
肋肋; 肋肋肋肋肋肋肋. 肋肋肋肋肋肋肋肋 了 了 了 了 了 了 1991
了 了 了 了 了 了 了 了 $x^n + y^n \neq z^n$ pour $n > 2$ et $x, y, z \neq 0$ 了 了
了 了 了 了 了 了 了 了 了 了 了 了 了 了 了 肋肋肋肋肋肋肋肋肋肋肋
肋肋肋肋肋肋肋肋肋肋肋肋肋肋

Remarque : Les commandes d'espacement, '`\ `' et `\kaispace`, produisent toutes les deux une case vide en mode rigide.

L'environnement kai dans un contexte mathématique

La commande `\kai` peut être utilisée à l'intérieur d'un environnement mathématique.

- **Exemples**

```
$$A=\pmatrix{
\kai ai412 /&\kai ai412 /&\kai ai411 ai411 /\cr
\kai ai412 /&\kai ai411 ai411 /&\kai ai412 /\cr
\kai ai411 ai411 /&\kai ai412 /&\kai ai412 /}$$
```

$$A = \begin{pmatrix} \text{隘} & \text{隘} & \text{艾艾} \\ \text{隘} & \text{艾艾} & \text{隘} \\ \text{艾艾} & \text{隘} & \text{隘} \end{pmatrix}$$

```
$$\kai a13 ai412 ai412 / \qquad
\cases{x=0&\kai ai412 ai412 ai412 ; /\cr
       x=1&\kai ai411 ai411 ai411 . /}$$
```

$$\text{锕隘隘} \qquad \begin{cases} x = 0 & \text{隘隘隘;} \\ x = 1 & \text{艾艾艾.} \end{cases}$$

- **Indices et exposants.** Pour éviter une saturation de mémoire, nous n'avons pas prévu le chargement automatique des tailles d'indice et sous-indice pour les caractères kai. Les commandes

  `\kaiind{<codes-kai>}` et `\kaiindind{<codes-kai>}`

  permettent pourtant de le faire manuellement. Par exemple :

  ```
  $\kai ai41 /_{\kaiind{ai412}^{\kaiindind{ai411}}}$
  ```

  donne 哎$_{隘^{艾}}$.

3. Écriture centrée

L'environnement d'écriture horizontale centrée est de la forme

```
\kaiC
    | <code-kai> | * |    cr     |    ,;:.?!~    |
    | 0123456789 | \ | \kaispace | text:<chaîne> |
/
```

et son mode d'utilisation est identique à celui de `\kai`.

```
\kaiC ai412 ai412 ai412 ai412 ai412 ai412 cr
{\setkaiintercharkern8pt}
ai41 ai41 ai41 ai41 ai41 ai41 ai41 cr \kainormal
ai412 ai412 ai412 ai412 ai412 ai412 ai412 /
```

隘隘隘隘隘隘隘
哎 哎 哎 哎 哎 哎 哎
隘隘隘隘隘隘隘

4. Écriture verticale

Nous proposons deux environnements d'écriture verticale

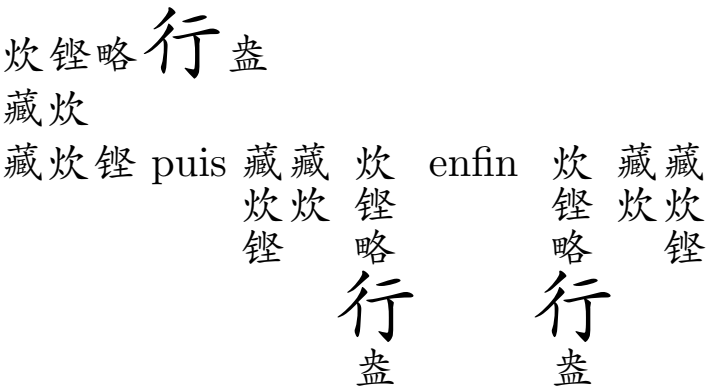
```
\kaiVLR ou \kaiVRL
    | <code-kai> | * |    cr     |    ,;:.?!~    |
    | 0123456789 | \ | \kaispace | text:<chaîne> |
/
```

(`V` pour vertical, `RL` pour "right to left" et `LR` pour "left to right" : les deux modes de juxtaposition des colonnes.)

Tout ce que nous avons dit à propos de `\kai.../` s'applique à ces nouveaux environnements. Il s'ensuit qu'il suffit de rajouter `VRL` ou `VLR` après une commande `\kai` pour faire pivoter la présentation. Par exemple la saisie :

```
\kai??? chui12 keng13 lve42 \kailarge hang24
        \kainormal ang41 cr cang22
        chui12 cr cang22 chui12 keng13 /
```

où l'on substituera, `kai???` par `\kai` puis `\kaiVRL` enfin `\kaiVLR`, donne :



Remarque : En mode vertical les caractères sur une colonne sont toujours centrés par rapport à sa ligne médiane.

Paramètres internes de l'environnement kaiV

Comme le mode horizontal, le mode vertical possède des paramètres internes :

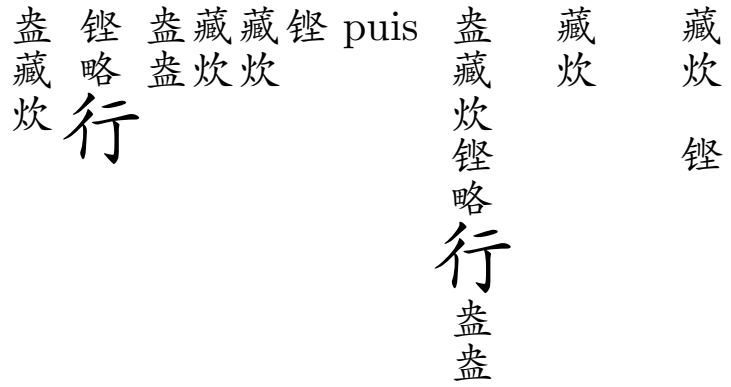
– `\kaicolwidth=<dim>` : largeur minimale d'une colonne. Lorsque TEX prépare une colonne, il détermine sa largeur naturelle et si

celle-ci est inférieure à `\kaicolwidth` elle est élargie jusqu'à égalité.

– `\kaiintercolkern=<dim>` : écart entre deux colonnes.
– `\kaimaxlines=<nombre>` : Nombre maximum des lignes par colonne ; par défaut 20.

Voici deux exemples :

```
\kaiVLR{\kaimaxlines=3}
        ang41 cang22 chui12 keng13 lve42 \kailarge
        hang24 \kainormal ang41 ang41 cr cang22
        chui12 cr cang22 chui12 \ keng13 /
        puis
\kaiVLR ang41 cang22 chui12 keng13 lve42 \kailarge
        hang24 \kainormal ang41 ang41 cr cang22
        {\kaicolwidth=1cm}
        chui12 cr cang22 chui12 \ keng13 /
```



Finesses de l'environnement vertical

La compilation d'un environnement vertical requiert des opérations très complexes qui le rendent particulièrement sensible et difficile à maîtriser lorsque l'on sort des sentiers battus. Les commandes suivantes s'avèrent de grand secours lorsque l'on est tenté d'utiliser cet environnement de manière inhabituelle.

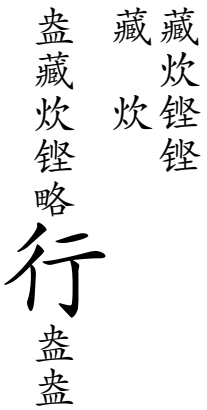
– `\kaiinsert{...}`. Insère une boîte dans la liste dont la largeur est sa largeur naturelle.

– `\kaiinsertsmash{...}`. Insère une boîte dans la liste dont la largeur est nulle.
– `\kaiinsertto{<dim>}{...}`. Insère une boîte dans la liste dont la largeur est `<dim>`.

Comme ces macros comportent au moins un paramètre, il faudra les entourer par des accolades à l'intérieur d'un environnement `kai`.

Exemple : Considérons la saisie :

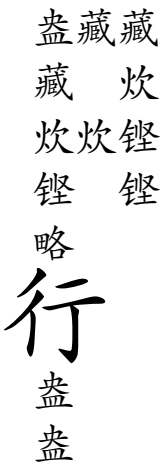
```
\kaiVLR ang41 cang22 chui12 keng13 lve42
        \kaiLarge hang24 \kainormal ang41  ang41
    cr cang22  \ chui12 cr cang22 chui12 keng13 keng13 /
```



et supposons que l'on veuille supprimer l'écart supplémentaire entre les deux premières colonnes. Comme celui-ci provient de la largeur plus importante du caractère en style `kaiLarge`, l'idée est de remplacer `\kaiLarge hang24` par `\kai\kaiLarge hang24 /` pour le mettre dans une boîte de largeur nulle. La saisie précédente devient alors

```
\kaiVLR ang41 cang22 chui12 keng13 lve42
 {\kaiinsertsmash{\kai\kaiLarge hang24 /}} ang41  ang41
     cr cang22  chui12 cr cang22 chui12 keng13 keng13 /
```

ce qui donne le résultat cherché :



5. Personnalisation des commandes kai

Si vous souhaitez modifier de manière permanente certains des paramètres internes des environnements `\kai`, `\kaiC`, `\kaiVRL` et `\kaiVLR` vous pouvez toujours définir des versions modifiées en posant au début de votre travail, par exemple :

```
\def\mykai{\kai{...}}
```

où `{...}` désigne la liste des commandes de modification des paramètres. Tout ce que nous avons dit à propos de `\kai.../` s'applique à l'environnement : `\mykai ... /`

6. Des macros qui posent des problèmes

Vous serez probablement obligé à un moment ou à un autre de mettre des macros TEX dans un environnement kai, et même parfois sans en être conscient. C'est le cas des caractères dits `actifs` qui représentent à eux seuls une macro TEX ; par exemple : dans les versions françaises des formats habituels et pour ce qui est les caractères `{ : ; , ? !}`. Ces macros étrangères au format kai peuvent poser de gros problèmes lors du traitement d'un environnement kai. Si vous vous retrouvez avec beaucoup de messages erreur de TEX, tentez toujours, en premier lieu, de faire précéder la macro ou le caractère suspect par la commande `\kaiprotect`, dont le but est d'inhiber le

développement de la commande (ou caractère actif) qui le suit dans les parties critiques de la compilation de l'environnement kai.

Exemple : la ligne `\kai\kailarge ; /` donne, dans ma version francisée du format `plain`, les messages d'erreur :

```
! Missing \endcsname inserted.
<to be read again>
                   \relax
;->\relax
          \ifmmode \string ; \else ...
\testifkaicommand ...dprefixe #1
                                    \end...
\binKai ...testifkaicommand {#1}
                                    \ifk...
l.475 \kai ;
                  /
```

La dernière ligne du message montre que le problème vient précisément du caractère ‘ ;’. Dans un tel cas, on doit avoir le réflexe de remplacer ‘ ;’ par un autre signe de ponctuation pour confirmer l'origine problème. Dès que vous trouvez l'élément suspect, n'hésitez pas à le faire précéder de la commande `\kaiprotect`. Dans le cas présent la ligne `\kai\kailarge {\kaiprotect ;} /` donne après compilation ‘;’ et ceci sans aucun message d'erreur.

Bien évidemment, le même type de recommandation s'applique aux macros de TEX.

7. Nouveaux styles de fontes Kai

La commande :

`\LoadStyleWeigthSize{<style>}{<gras>}{<taille>}`

charge l'ensemble de fontes Kai dans la taille et le gras spécifiés. Le paramètre `<gras>` concerne les variantes de gras et n'a que les deux valeurs possible : la chaîne de caractères vide pour la version normale et `Bold` pour la version grasse. L'accès aux nouvelles fontes

se fait alors par l'intermédiaire de la commande dont le nom est `\<style>`.

Le format kai comporte la commande :

`\LoadStyleWeigthSize{kaibf}{Bold}{}`

de sorte que `\kai ao210 \kaibf lou25 ling217 /` donne 鏖耧菱.

Remarque : Lorsque le paramètre `<taille>` n'est pas spécifié, c'est la taille courante qui est prise en compte.

Autre exemple :

`\LoadStyleWeigthSize{kaiLarge}{}{40pt}` (∗)

crée le style `kaiLarge` comportant les fontes kai à 40 points et déclare `\kaiLarge` comme le raccourci permettant d'y accéder. Les commandes suivantes montrent le mode d'utilisation après (∗).

`\kai ao210 lou25 ling217 /` donne : 鏖耧菱 et

`\kai\kaiLarge ao210 lou25 ling217 /` :

鏖耧菱

Style par défaut

`\kainormal` est le raccourci du style par défaut ; exemple :

`\kai\kaiLarge ao210 \kainormal lou25 \kaibf ling217 /`

donne :

鏖耧菱 (‡)

8. Déclaration virtuelle de nouvelles tailles

Le défaut de `\LoadStyleWeigthSize` est qu'elle charge une quarantaine de fontes en une seule fois et que cela pourrait rapidement saturer la mémoire de la machine. La variante

`\kainewstyle{<style>}{<taille>}`,

crée les styles : `\<style>` et `\<style>bf`, formes respectivement normal et grasse des fontes Kai à la taille `<taille>`, mais ne charge une fonte que lorsque c'est nécessaire. Il est possible que cette commande entraîne un leger ralentissement de la compilation.

La ligne

```
\kainewstyle{kailarge}{20pt}
```

a été déclarée au début de ce mode d'emploi. On trouve alors dans le `log` de compilation les commentaires suivants.

```
[7]
------>Loading font kailargeKF29=kai29 at20.0pt
------>Loading font kailargeKFrm=cmr10  at20.0pt
[8] [9] [10] [11]
------>Loading font kailargeKF25=kai25 at20.0pt
```

Exemple d'utilisation :

```
  \kaiC\kailarge    ao210 lou25 2000 ling217 cr
       \kailargebf  ao210 lou25 2000 ling217 /
```

donne :

慶耬2000菱
慶耬2000菱

9. Caractères isolés dans de nouvelles tailles

Lorsque peu de caractères seront nécessaires dans une nouvelle taille taille, il faut privilégier les commandes

```
\kaiatsize{<dim>}{<code-kai>...<code-kai>}
\kaiatsizebf{<dim>}{<code-kai>...<code-kai>}
\kaiscaled{pour mille}{<code-kai>...<code-kai>}
\kaiscaledbf{pour mille}{<code-kai>...<code-kai>}
```

qui chargent uniquement les fontes contenant les caractères kai des codes `<code-kai>...<code-kai>` dans la taille spécifiée sans créer de style spécifique.

- **Exemples :**
  - `$$\kaiatsize{10cm}{ling217}$$`, donne `ling217` à `10cm` (en format A4) et centré :

  - pour afficher un caractère kai à une dimension proportionnelle (en pour mille) à la taille du style courant :
    `\kai\kaibf cong17 / \kaiscaledbf{1000}{cong17} \kaiscaled{1500}{cong17} \kaiscaledbf{2000}{cong17} \kaiscaled{500}{cong17} \kaiscaled{100}{cong17}`
    donnent respectivement :

聰 聰 聰 **聰** 聰 聰

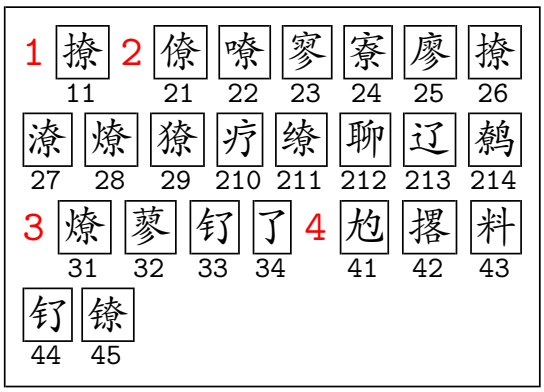
Remarque : Les commandes `\kaiind` et `\kaiindind` sont définies dans le format kai par :

```
\def\kaiind#1{\kaiscaled{707}{#1}}
\def\kaiindind#1{\kaiscaled{500}{#1}}
```

10. A propos de l'aide en ligne

La manière dont les codes incomplets présentent les échantillons des caractères dépend de l'endroit où ils sont saisis. Par exemple, une saisie de la forme : `\centerline{\kai ai412 li310 liao /}` donnera une présentation, dite "flottante" :

隘醴(liao)



(*)

bien différente de celle de la page 3.

Comme les présentations flottantes peuvent également poser des problèmes dûs notamment à un bouleversement de la mise en page, nous avons prévu un mode d'aide *différée* que l'on active par la commande `\kaipostponehelp`. Lorsque cette dernière commande est présente, l'exemple précédent donne uniquement :

隘醴(liao)◇

et de manière analogue pour

```
\centerline{\kai ai412 li310 liao3 /}
```

qui donne :

隘醴(liao3)◇

On aura observé l'apparition du signe '◇' (rouge à l'écran). Celui-ci indique que le mode différé est actif et que l'aide a été affectée à la commande `\kaihelp`. Nous conseillons de placer cette commande en fin de travail pour éviter des perturbations importantes de mise en page. Le signe '◇' est un lien hypertexte sur la page d'aide et il

suffit de double cliquer sur ce signe pour être aussitôt amené à la page en question ; on en revient par le même procédé.

On peut également placer `\kaihelp` à l'endroit le plus simple possible après la zone de saisie, par exemple au premier point se trouvant en mode vertical externe de TEX. Dans l'exemple précédent cela revient à saisir

```
\centerline{\kai ai412 li310 liao /}\kaihelp
```

ce qui donne :

隘醴 (liao)◇

(liao1) 撩 11

(liao2) 僚 21 嘹 22 寥 23 寮 24 廖 25 撩 26 潦 27 燎 28 獠 29 疗 210 缭 211 聊 212 辽 213 鹩 214

(liao3) 燎 31 蓼 32 钌 33 了 34

(liao4) 尥 41 撂 42 料 43 钌 44 镣 45

Les entêtes des lignes d'aide sont également des liens hypertextes ; ils ramènent à l'endroit où se trouve le code kai incomplet ayant déclenché l'aide en ligne, bien évidemment cette fonctionnalité n'est vraiment utile que lorsque l'on reporte la commande `\kaihelp` en fin de travail.

Remarque : La commande `\kaihelp` équivaut à la commande `\relax` de TEX lorsque elle ne présente aucune aide en ligne. En particulier, elle sera inerte dans votre fichier source après complétion des codes kai qui la précédent ; il n'est donc pas nécessaire de l'enlever.

Remarque : L'aide en ligne différée cumule toutes les demandes d'assistance non résolues dans la commande `\kaihelp`, ce qui donne une autre raison de la placer en fin de travail.

Remarque : Le mode différée est l'unique mode d'aide en ligne dans

les environnements kai verticaux et les environnements TEX mathématiques.

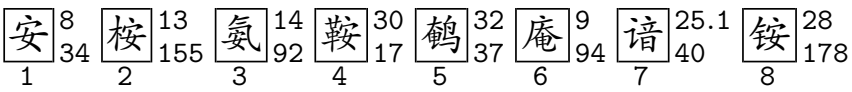
Remarque : La commande `\kaiinlinehelp` désactive le mode d'aide différée.

Affichage des paramètres internes d'un caractère kai

En dehors des trois paramètres `<syllabe><accent><indice>` constituant le code kai d'un caractère chinois, il en existe deux autres qui désignent respectivement la fonte kai qui le contient et sa position dans celle-ci (comprise entre 0 et 255).

- La commande `\kaiallinfotrue` active l'affichage de paramètres internes dans les aides en ligne.

  La commande `{\kaiallinfotrue\kai an1 /}` donne :

  (an1) 

  où l'exposant (`exp`) désigne la fonte kai qui contient le caractère et l'indice (`ind`) est sa position dans la fonte en question.

  Remarque : La commande `\thekaichar{<exp>}{<ind>}` est la forme la plus robuste et rapide pour afficher un caractère kai dans le style courant. (En mathématiques, cette commande devra être à l'intérieur d'une boite, p.ex. :

  `\hbox{\thekaichar{<exp>}{<ind>}}`.)

  Remarque : La commande `\kaiallinfotrue` est inopérante dans les présentations flottantes (∗).

- Lorsque l'on cherche à connaître les paramètres internes d'un suite de codes déjà saisie, par exemple de :

  `\kaiC ang41 chao11 cr dia31 gong34 fu235 /,`

  il suffit de rajouter la commande `\kaishowparams` à l'intérieur de l'environnement. La saisie :
  `\kaiC\kaishowparams ang41 chao11 cr dia31 gong34 fu235 /`
  donne alors :

盎 18 7 劓 3 153
ang41 chao11

嗲 5.1 73 珙 16 136 鞸 22 231
dia31 gong34 fu235

11. Catalogue

La commande `\kaicatalog` imprime la liste de tous les caractères disponibles classés par syllabes et accents ; en voici un extrait pour les cinq premières syllabes :

a0 : 啊 01

a1 : 啊 11 腌 12 锕 13 阿 14

a2 : 啊 21

a3 : 啊 31

a4 : 啊 41

---

ai1 : 哀 11 哎 12 唉 13 嗳 14 埃 15 挨 16 锿 17

ai2 : 挨 21 捱 22 癌 23

ai3 : 哎 31 嗳 32 矮 33 蔼 34 霭 35

ai4 : 哎 41 唉 42 嗌 43 嗳 44 嫒 45 暧 46 爱 47 瑷 48 砹 49 碍 410 艾 411 隘 412

---

an1 : 安 11 按 12 氨 13 鞍 14 鹌 15 庵 16 谙 17 铵 18

an3 : 埯 31 揞 32

an4 : 岸 41 按 42 案 43 珙 44 黯 45 暗 46

---

ang1 : 肮 11

ang2 : 昂 21

ang4 : 盎 41

---

ao1 : 凹 11

ao2 : 嗷 21 廒 22 敖 23 熬 24 獒 25 翱 26 聱 27 螯 28 遨 29 鏖 210 鳌 211

ao3 : 媪 31 拗 32 袄 33

ao4 : 傲 41 坳 42 奥 43 岙 44 懊 45 拗 46 澳 47 鏊 48 骜 49

---

et lorsque `\kaiallinfotrue` est déclarée avant `\kaicatalog`, on obtient :

a0 : 啊 01 5.1 2

a1 : 啊 11 5.1 2 腌 12 22 124 锕 13 28.1 9 阿 14 29 63

a2 : 啊 21 5.1 2

a3 : 啊 31 5.1 2

a4 : 啊 41 5.1 2

---

ai1 : 哀 11 5 133 哎 12 5 143 唉 13 5 175 嗳 14 5.1 74 埃 15 6 100 挨 16 11 154 锿 17 28.1 46

ai2 : 挨 21 11 154 捱 22 11 183 癌 23 17 155

ai3 : 哎 31 5 143 嗳 32 5.1 74 矮 33 18 105 蔼 34 23.1 89 霭 35 29 151

ai4 : 哎 41 5 143 唉 42 5 175 嗌 43 5.1 53 嗳 44 5.1 74 嫒 45 7 159 暧 46 12 140 爱 47 15 116 瑷 48 16 178 砹 49 18 139 碍 410 18 165

艾 23 3 411　隘 29 102 412

---

an1： 安 8 34 11　桉 13 155 12　氨 14 92 13　鞍 30 17 14　鹌 32 37 15　庵 9 94 16　谙 25.1 40 17　铵 28 178 18

an3： 埯 6 115 31　揞 11.1 29 32

an4： 岸 8 171 41　按 11 139 42　案 13 154 43　玵 16 136 44　黯 32 95 45　暗 12 138 46

---

ang1： 肮 22 48 11

ang2： 昂 12 82 21

ang4： 盎 18 7 41

---

ao1： 凹 3 90 11

ao2： 嗷 5.1 76 21　廒 9 103 22　敖 12 16 23　熬 15 97 24　獒 16 100 25　翱 21 201 26　聱 22 21 27　螯 24 135 28　遨 27 156 29　鏖 28 72 210　鳌 31 152 211

ao3： 媪 7 149 31　拗 11 108 32　袄 25 23 33

ao4： 傲 2 212 41　坳 6 70 42　奥 7 53 43　岙 8 157 44　懊 10 224 45　拗 11 108 46　澳 14.2 42 47　鏊 28 71 48　鳌 31 40 49

Paris, Jeudi 15 juin 2000