Un environnement pour composer du texte sur plusieurs colonnes.*†‡

Frank MITTELBACH§
mél : cf. au début du fichier source

Imprimé le 31 octobre 2001

Résumé

Ce document décrit l'utilisation et l'implémentation de l'environnement multicols. Cet environnement permet de passer d'un format sur une colonne à un format sur plusieurs colonnes et inversement, sur la même page. Les notes de bas de page sont traitées correctement (dans la plupart des cas), mais seront placées en bas de la page et non en bas de chaque colonne. Le mécanisme des flottants de LaTeX est cependant partiellement désactivé dans la réalisation actuelle. et sera ajouté dans une version future. Pour le moment, seuls les flottants portant sur toute la largeur de la page (c'est-à-dire les formes ∗) peuvent être utilisés à l'intérieur de la portée de l'environnement.

Préface pour la version 1.5

Cette nouvelle version contient deux modifications principales : multicols prendra maintenant en compte jusqu'à 10 colonnes et deux possibilités supplémentaires de réglage ont été ajoutées à la routine d'équilibrage. Cette routine vérifie maintenant la mauvaise qualité des colonnes résultantes et rejette les résultats des calculs qui sont supérieurs à un seuil donné.

En même temps, multicols a été mis à jour pour fonctionner avec LaTeX 2_ε.

Je[1] vous présente mes excuses pour le niveau de documentation du code, mais mes responsabilités envers LaTeX 2_ε m'occupent trop pour faire du bon travail. J'espère que cela sera corrigé dans un proche avenir.

1 Introduction

Passer d'une à deux colonnes et inversement est possible en LaTeX, mais chaque utilisation de `\twocolumn` ou de `\onecolumn` débute une nouvelle page. De plus, la toute dernière page composée en deux colonnes n'est pas équilibrée et cela donne souvent une colonne de droite vide ou presque vide. Quand j'ai commencé à écrire des macros pour doc.sty (cf. « The doc–Option », *TUGboat* volume 10 #2, pp. 245–273), je pensais qu'il serait mieux de placer l'index sur la même page que la bibliographie. L'équilibrage de cette dernière page n'est pas seulement une simple question d'esthétique, cela gagne aussi de la place, à condition, bien sûr, qu'il soit possible de commencer la section suivante sur la même page. Comparativement, réécrire l'environnement d'index est facile, mais le but étant de construire un environnement qui prenne en compte les notes de bas de page, les flottants..., c'était une tâche plus ardue. Cela m'a pris un week-end entier[2] pour réunir les quelques lignes de code ci-dessous et il y a encore de

*Version : v1.5w, dernière révision : 1999/10/21

†Note : Ce package est distribué sous des conditions qui influent sur son utilisation dans un contexte commercial. Cf. les détails au début du fichier source.

‡Titre original : An environment for multicolumn output

§Traduction française par Jean-Pierre DRUCBERT et Françoise MARRE-FOURNIER, du 06/03/2000.

[1] Frank MITTELBACH

[2] J'avais commencé avec l'algorithme présenté page 417 du TeXbook. Sans cette aide, un week-end n'aurait pas été suffisant.

grandes chances que j'aie malgré tout oublié quelque chose.

Essayez-le et adoptez-le et, *s'il vous plaît*, envoyez vos suggestions et signalez les bugs directement à Mainz.

2 Interface utilisateur

Pour utiliser cet environnement, il suffit de dire simplement

\begin{multicols}{⟨*number*⟩}
⟨*multicolumn text*⟩
\end{multicols}

où ⟨*number*⟩ est le nombre demandé de colonnes et ⟨*multicolumn text*⟩ peut contenir des commandes LaTeX arbitraires, sauf que les flottants et les notes marginales ne sont pas autorisés dans la version actuelle[3].

Comme première action, l'environnement multicols mesure la page courante pour déterminer s'il y a assez de place pour une certaine portion de sortie sur plusieurs colonnes. Ceci est contrôlé par la variable ⟨*dimen*⟩ **\premulticols** dont la valeur peut être modifiée par les commandes LaTeX.

Si l'espace est inférieur à **\premulticols**, une nouvelle page est commencée. Sinon, un saut vertical **\vskip** de hauteur **\multicolsep** est ajouté[4].

Lorsque la fin de l'environnement multicols est atteint, un mécanisme analogue est employé, mais nous testons alors si un espace supérieur à **\postmulticols** est disponible. De nouveau nous ajoutons l'espace **\multicolsep** ou commençons une nouvelle page.

Il est souvent pratique de placer du texte en chapeau au-dessus de toutes les colonnes, avant la sortie en multi-colonnes, sans qu'une rupture de page ne puisse se produire entre ce chapeau et ces colonnes. Pour réaliser cela, l'environnement multicols a un second argument optionnel qui peut être utilisé dans ce but. Par exemple, le texte que vous lisez actuellement a commencé par :

```
\begin{multicols}{3}
[\section{Interface
        utilisateur}]...
```

Si un tel texte est inhabituellement long (ou court) la valeur de **\premulticols** peut avoir besoin d'être ajustée pour éviter une mauvaise rupture de page. Nous offrons donc un troisième argument qui peut être utilisé pour outrepasser la valeur par défaut de **\premulticols** seulement pour cette occasion. Donc si vous voulez combiner un long texte sur une seule colonne avec un environnement multicols, vous pourriez écrire :

```
\begin{multicols}{3}
    [\section{Index}
     Cet index contient ...]
    [6cm]
 ...
```

L'espace entre les colonnes est contrôlé par le paramètre de longueur **\columnsep**. La largeur des colonnes individuelles est calculée automatiquement à partir de ce paramètre et de la valeur courante de **\linewidth**. Ici une valeur de 18.0pt a été utilisée.

La séparation des colonnes par des filets verticaux est obtenue en forçant le paramètre **\columnseprule** à une longueur positive. Si cette longueur est nulle ou négative, le filet est invisible. Ici elle vaut 0.4 pt.

Puisque l'espacement des lignes dans des colonnes étroites a tendance à devoir être ajusté, nous fournissons aussi un paramètre ⟨*skip*⟩ (longueur élastique) nommé **\multicolbaselineskip** qui sera ajouté au paramètre **\baselineskip** à l'intérieur de chaque environnement multicols. Il est prudent d'utiliser ce paramètre avec précaution ou de ne pas y toucher ; il n'est destiné qu'aux concepteurs de packages car même de toutes petites modifications peuvent produire des bouleversements inattendus dans votre document.

2.1 Équilibrage des colonnes

Outre les paramètres mentionnés ci-dessus, d'autres sont disponibles pour influencer la mise en forme des colonnes produites.

Le traitement des paragraphes en TeX est contrôlé par plusieurs paramètres. L'un des plus importants est appelé **\tolerance** : il contrôle la « *looseness* » (c'est-à-dire la quantité d'espace blanc entre les mots). Sa valeur par défaut est 200 (la commande **\fussy** de LaTeX), qui est trop faible pour des colonnes étroites. D'autre part, la déclaration **\sloppy** (qui force **\tolerance** à $10000 = \infty$) est trop grande,

[3] Ceci est imposé par un manque de temps. Pour autoriser les flottants, il faudra réécrire toute la routine de sortie de LaTeX.

[4] En fait l'espace ajouté peut être plus petit car la commande \addvspace est utilisée. (cf. le manuel LaTeX pour plus d'informations sur cette commande)

[5] Regardez le paragraphe suivant, qui a été composé avec la déclaration \sloppy

permettant des espacements vraiment trop laids [5].

Nous utilisons donc un autre paramètre `\multicolstolerance` pour la valeur de `\tolerance` à l'intérieur de l'environnement multicols. Sa valeur par défaut est 9999, qui est inférieure à l'infini mais assez « mauvaise » pour la plupart des paragraphes dans un environnement multicolonne. Changer sa valeur doit être fait *en dehors* de l'environnement multicols. Puisque `\tolerance` est forcée à `\multicolstolerance` au début de chaque environnement multicols, on peut localement outrepasser cette valeur par défaut en imposant `\tolerance␣=␣`⟨*valeur désirée*⟩. Il existe également un paramètre `\multicolpretolerance` conservant la valeur de `\pretolerance` à l'intérieur d'un environnement multicols. Ces deux paramètres ne sont habituellement utilisés que par les concepteurs de packages.

La génération de la sortie sur plusieurs colonnes peut être divisée en deux parties. Dans la première partie, nous rassemblons du matériel pour la page, l'envoyons vers la sortie, collectons du matériel pour la page suivante, et ainsi de suite. Dans une seconde étape, l'équilibrage sera fait lorsque la fin de l'environnement multicols sera atteint. Dans la première étape TEX pourrait considérer davantage de matériel en trouvant les colonnes finales qu'il n'en utilise effectivement lors de l'émission de la page. Ceci peut provoquer un problème si une note en bas de page est trouvée dans la partie considérée, mais non utilisée sur la page courante. Dans ce cas la note peut apparaître sur la page courante, tandis que l'appel de note correspondant peut se trouver sur la page suivante [6] Donc l'environnement multicols donne un message d'avertissement [7] chaque fois qu'il est incapable d'utiliser en totalité le matériel considéré jusque là.

Si vous n'utilisez pas de notes trop souvent, les risques que quelque chose se passe mal sont très faibles, mais si cela arrive, vous pouvez aider TEX en plaçant une commande `\pagebreak` dans le document final. Un autre moyen pour influencer le comportement de TEX à ce sujet est offert par la variable compteur « collectmore ». Si vous utilisez la déclaration `\setcounter` pour forcer ce compteur à un ⟨*nombre*⟩, TEX considèrera ⟨*nombre*⟩ lignes en plus (ou en moins) avant de prendre sa décision finale. La valeur -1 pourra donc résoudre tous vos problèmes au prix de colonnes un peu moins optimales.

Dans la seconde étape (équilibrage des colonnes) nous avons d'autres gadgets. D'abord, vous pouvez dire `\raggedcolumns` si vous n'avez pas besoin que les lignes du bas des colonnes soient alignées. L'option `\flushcolumns` est prise par défaut, donc TEX essaiera d'aligner les lignes de base du haut et du bas de toutes les colonnes.

De plus, vous pouvez utiliser un autre compteur, « unbalance », en lui donnant une valeur positive ⟨*nombre*⟩. Ceci rendra toutes les colonnes, sauf celle la plus à droite, plus longues de ⟨*nombre*⟩ lignes qu'elles ne l'auraient été normalement. Les « lignes » dans ce contexte sont des lignes de texte normal (c'est-à-dire écartées de `\baselineskip`) ; ainsi, si votre colonne contient des hors-texte, par exemple, vous aurez besoin d'un ⟨*nombre*⟩ plus grand pour décaler quelque chose d'une colonne vers une autre.

Contrairement à « collectmore », le compteur « unbalance » est remis à zéro à la fin de l'environnement afin qu'il ne s'applique qu'à un seul environnement multicols.

Les deux méthodes peuvent être combinées mais je conseille de n'utiliser ces possibilités que pour mettre la dernière touche à d'importantes publications.

Deux nouvelles possibilités de réglage ont été ajoutées dans la version 1.5. TEX permet de mesurer la mauvaise qualité – la laideur – d'une colonne sous forme d'une valeur entière, où 0 signifie optimal et toute autre valeur plus haute dénote la présence d'une certaine quantité d'espace blanc en trop. 10000 est considéré comme infiniment mauvais (TEX ne fait plus de distinction au-dessus de cette valeur). De plus, la valeur spéciale 100000 signifie un débordement (c'est-à-dire que la colonne contient plus de texte qu'il n'est possible d'y mettre).

La nouvelle version mesure désormais chaque colonne engendrée et ignore les solutions pour lesquelles au moins une colonne a une laideur supérieure à la valeur du compteur columnbadness. La valeur par défaut pour ce compteur est 10000, donc TEX acceptera toutes les solutions sauf celles avec débordement. En donnant à ce ce compteur une valeur plus petite, vous pouvez forcer l'algorithme à rechercher des solutions qui n'auront pas de colonnes contenant beaucoup d'es-

[6] La raison qui se cache derrière ce comportement est le caractère asynchrone du *page_builder* de TEX. Cependant ceci pourrait être évité en définissant des routines de sortie très compliquées qui n'utilisent pas des primitives TEX, telles que `\insert`, mais font tout à la main. Ceci n'a hélas pas été fait par manque de temps. Cela dépasse vraiment les limites d'un week-end.

[7] et ce message sera émis même s'il n'y a pas de notes dans cette partie du texte

pace blanc.

Cependant, si la valeur choisie est trop basse, l'algorithme peut ne trouver aucune solution acceptable et finalement il choisira la solution extrême qui consiste à placer tout le texte dans la première colonne.

Souvent, lorsque les colonnes sont équilibrées, il est impossible de trouver une solution qui distribue le texte équitablement entre toutes les colonnes. Si c'est le cas, habituellement la dernière colonne a moins de texte que les autres. Dans les premières versions, ce texte était étiré pour produire une colonne de même hauteur que toutes les autres, donnant parfois des colonnes vraiment horribles à voir.

Dans la nouvelle version, cet étirement n'est effectué que si la laideur de la dernière colonne n'est pas plus grande que la valeur du compteur finalcolumnbadness. Sa valeur par défaut est 9999, ce qui empêche l'étirement pour toutes les colonnes que TeX considérerait comme infiniment laides. Dans ce cas, la colonne finale peut être plus courte, ce qui donne un bien meilleur résultat.

Il y a deux autres paramètres qui sont encore un peu expérimentaux : l'un s'appelle `\multicolovershoot` et l'autre `\multicolundershoot`. Ils contrôlent la quantité d'espace dont une colonne est autorisée à être « trop pleine » ou « trop courte » sans affecter la laideur de la colonne. Ils valent 2pt par défaut.

2.2 Non équilibrage des colonnes

Bien que ce package ait été écrit pour résoudre le problème de l'équilibrage des colonnes, il m'a souvent été demandé une version dans laquelle tout l'espace blanc est automatiquement placé dans la (les) dernière(s) colonne(s). Depuis la version v1.5q ceci existe désormais : si vous utilisez `multicols*` au lieu de l'environnement habituel, les colonnes sur la dernière page ne sont pas équilibrées. Bien sûr, cet environnement ne fonctionne qu'au niveau le plus haut, c'est-à-dire à l'intérieur d'une boîte qu'il faut équilibrer afin de déterminer une hauteur de colonne en l'absence de valeur fixée.

2.3 Coupure manuelle des colonnes

Un autre demande souvent entendue était : « Comment puis-je dire à LaTeX qu'il doit couper la première colonne après cette ligne-là ? ». La commande `\pagebreak` (qui fonctionne avec l'option `twocolumn` de LaTeX) ne sert à rien ici car elle terminerait la phase de collecte de multicols et donc toutes les colonnes sur cette page. Dans la version 1.5u la commande `\columnbreak` a donc été ajoutée. Si elle est utilisée à l'intérieur d'un paragraphe, elle marque la fin de la ligne courante comme point de coupure de colonne souhaité. Vous pouvez observer son effet sur la première page[8] la page où quelques lignes de texte ont été artificiellement forcées dans la troisième colonne (ce qui a donné un peu d'espace blanc entre les paragraphes dans la deuxième colonne).

2.4 Éléments flottants à l'intérieur d'un environnement multicols

À l'intérieur de l'environnement multicols, les commandes habituelles d'éléments flottants sont disponibles (sous leur forme $*$) mais leur fonction est assez différente de celle qu'elles avaient dans le mode deux colonnes du LaTeX standard. Les flottants $*$, comme `figure*`, indiquent des éléments flottants sur toute la largeur de la page qui sont traités d'une manière analogue à celle des flottants normaux en dehors de l'environnement multicols. Cependant, il ne seront jamais imprimés sur la page où ils sont rencontrés. En d'autres termes, on peut influencer leur positionnement en spécifiant une combinaison de `t`, `b` et/ou `p` dans leur argument optionnel, mais `h` ne fonctionne pas car le premier endroit possible est le haut de la page suivante. Il faut aussi noter que ceci signifie que le comportement de leur placement est déterminé par les valeurs de `\topfraction`, etc. et non par `\dbl`...

2.5 Avertissements

Dans certaines circonstances, l'utilisation de l'environnement multicols peut provoquer quelques avertissements de la part de TeX ou LaTeX. Voici une liste des plus importants et des causes possibles

`Underfull \hbox (badness ...)`
: Comme les colonnes sont souvent très étroites, TeX n'a pas été capable de trouver un bonne façon de couper le paragraphe en lignes. « Underfull » signale une ligne creuse mais tant que la laideur est en dessous de 10000 le résultat est probablement acceptable.

`Underfull \vbox ... while \output is active`
: Si une colonne contient un caractère ayant une profondeur

[8] Dans le fichier original, la rupture de colonne était réalisée à la fin de la première colonne de la page 3. Cependant, pour respecter l'esprit du package multicols, cette rupture de page à été déplacée dans le § de préface. NdT

inhabituelle, par exemple une « (», dans la ligne du bas, alors ce message peut apparaître. Il n'est en général pas significatif tant que la valeur ne dépasse pas quelques points.

`LaTeX Warning : I moved some lines to the next page`
Comme cela a été dit plus haut, multicols perturbe parfois la numérotation des notes infrapaginales. Par précaution, chaque fois qu'il y a une note sur une page pour laquelle multicols a dû laisser un reste pour la page suivante, cet avertissement apparaît. Vérifiez la numérotation des notes infrapaginales sur cette page. S'il se trouve qu'elle est incorrecte, il vous faudra couper manuellement la page en utilisant `\newpage` ou `\pagebreak[...]`.

`Floats and marginpars not allowed inside 'multicols' environment !`
Ce message apparaît si vous essayez d'utiliser une commande `\marginpar` ou une forme sans étoile d'un environnement figure ou table. De tels éléments flottants disparaîtront !

2.6 Tracer la sortie

Pour comprendre le raisonnement qu'il y a derrière les décisions que prend TEX lors du traitement d'un environnement multicols, un mécanisme de traçage est offert. Si vous donnez une valeur positive ⟨*nombre*⟩ au compteur « multicols », vous obtiendrez des informations de traçage à la fois sur votre terminal et dans le fichier `.log`

⟨*nombre*⟩ = 1. TEX vous dira, chaque fois qu'il entre ou sort de l'environnement multicols, le nombre de colonnes sur lequel il travaille et sa décision s'il faut faire une nouvelle page avant ou après l'environnement ;

⟨*nombre*⟩ = 2. Dans ce cas, vous aurez en plus des informations issues de la routine d'équilibrage : les hauteurs essayées pour les colonnes les plus à gauche et à droite, des informations sur le rétrécissement si la déclaration `\raggedcolumns` est effective ainsi que la valeur du compteur « unbalance » si elle est positive.

⟨*nombre*⟩ = 3. En donnant cette valeur au ⟨*nombre*⟩, l'algorithme de gestion de marques sera aussi tracé. Cela montrera quelles marques sont trouvées, quelles marques sont prises en considération, etc. Pour comprendre pleinement ces informations, vous aurez probablement besoin de lire avec soin l'implémentation.

⟨*number*⟩ ≥ 4. Donner une valeur aussi élevée fera qu'en plus une `\hrule` sera placée dans la sortie, pour séparer la partie de texte qui a déjà été considérée sur la page précédente du reste. Évidemment cette valeur *ne doit pas* être utilisée pour l'impression définitive. Elle activera aussi beaucoup plus de code de correction des erreurs pour la prise en compte des marques.

3 Préface des anciennes versions

3.1 Preface de la version 1.4

À côté de la correction de bugs, comme indiqué dans le fichier multicol.bug, cette nouvelle version améliore l'environnement en autorisant l'équilibrage dans des contextes arbitraires. Il est maintenant possible, par exemple, d'équilibrer un texte à l'intérieur d'un multicols ou d'une minipage comme montré en 2, où un environnement multicols est utilisé à l'intérieur d'un environnement quote. Maintenant, il est même possible d'imbriquer des environnements multicols.

La seule restriction pour un tel environnement multicols intérieur (imbriqué ou dans le mode vertical interne de TEX) est que de telles variations produiront une boîte dont le matériel y sera équilibré, et donc ne pourront être à cheval sur plusieurs pages ou colonnes.

De plus, j'ai réécrit l'algorithme d'équilibrage pour qu'il produise maintenant des résultats légèrement meilleurs.

J'ai mis à jour la documentation mais excusez-moi par avance pour les parties « laissées de côté » qui sont passées à travers les corrections.

Une note pour les utilisateurs qui désirent améliorer l'algorithme d'équilibrage de multicols : la routine d'équilibrage est maintenant constituée par une seule macro appelée `\balance@columns`. Cela signifie que l'on peut aisément essayer différentes routine d'équilibrage en réécrivant cette macro. L'interface d'utilisation est expliquée dans le tableau 1. Il y a plusieurs améliorations possibles, on peut penser à l'intégration de la fonction `\badness` de TEX3, définir un algorithme plus rapide de recherche de la bonne hauteur de colonne, etc. Si quelqu'un pense qu'il/elle a un solution d'amélioration, je serais heureux de le savoir. Mais, s'il vous plait, res-

La macro \balance@columns qui contient le code pour équilibrer le matériel rassemblé, est une macro sans paramètres. Elle suppose que le matèriel à équilibrer est stocké dans la boîte \mult@box qui est une \vbox. Elle « connait » aussi tous les paramètres fixés par l'environnement multicols, comme \col@number, etc. Elle peut aussi supposer que \@colroom est l'espace encore disponible dans la page en cours. Quand elle a fini, elle doit retourner les colonnes individuelles dans des boîtes appropriées pour la suite du traitement par \page@sofar. Cela signifie que la colonne gauche devra être stockée dans la boîte registre \mult@gfirstbox, la suivante dans le registre \mult@firstbox + 2, ..., seule la dernière, par exception, dans le registre \mult@grightbox. En outre, elle doit fixer les deux macros \kept@firstmark et \kept@botmark pour récupérer les valeurs des marques initiales et du bas comme trouvées dans les colonnes individuelles. Il y a quelques fonctions d'aide dans la section 5.1 qui peuvent être utilisées pour cela. Donner les marques correctes « à la main » n'est pas trivial et il peut être nécessaire de jeter un coup d'œil à la documentation et à l'implémentation de \balance@columns avant d'essayer à nouveau.

TAB. 1 – Description de l'interface de \balance@columns

pectez le copyright et de changez pas multicol.dtx directement !

3.2 Préface de la version 1.2

Après que l'article sur l'environnement multicols a été publié dans *TUGboat* 10#3, j'ai reçu de nombreuses demandes pour ces macros. Quoi qu'il en soit, j'ai aussi reçu une version modifiée de mon fichier style, en même temps qu'une lettre me demandant si je voulais inclure les changements pour donner de meilleurs résultats de mise en forme des paragraphes dans les cas de lignes étroites. Les principales différences avec mon fichier style original étaient des paramètres additionnels (comme \multicoladjdemerits à utiliser pour \adjdemerits, etc) qui pouvaient influencer l'algorithme de rupture de lignes.

Mais, en fait, remettre à zéro de tels paramètres, ou même pire leur donner une valeur négative, ne donnera pas de meilleures coupures de lignes dans l'environnement multicols. L'algorithme de rupture de ligne de TeX n'envisagera que les coupures de lignes potentielles qui peuvent être réalisées sans induire une laideur supérieure à la valeur courante de \tolerance (ou \pretolerance pour la première passe). Si ce n'est pas possible, alors, en dernier ressort, TeX produira des boîtes débordantes. Tous ces points de rupture possibles (et seulement ceux-la) seront pris en compte et, finalement, la séquence qui sera la moins pénalisante sera choisie. Cela signifie qu'une valeur de −1000 pour \adjdemerits enjoint à TeX de préférer des lignes visiblement incompatibles plutôt que de produire de meilleures ruptures de lignes.

Cependant, avec TeX 3.0, il est possible de produire des ruptures de lignes convenables même dans de petites colonnes, en fixant \emergencystretch à une valeur appropriée. J'ai implémenté une version qui est capable de fonctionner à la fois pour l'ancien et le nouveau TeX (en fait, elle ignorera simplement la nouvelle caractéristique si elle n'est pas disponible). Le calcul de \emergencystretch est probablement incorrect. J'ai fait quelques tests, mais, bien sûr, on aura beaucoup plus d'expérience avec les nouvelles possibilités pour atteindre la qualité maximum.

La version 1.1a a une charmante « caractéristique » : la punition pour l'utilisation des flottants interdits était leur suppression définitive à partir de la \@freelist de LaTeX, de telle sorte qu'après quelques \marginpar dans l'environnement multicols, les flottants étaient désactivés pour toujours (Merci à Chris ROWLEY pour l'avoir remarqué). J'ai enlevé ce dysfonctionnement et, en même temps, décidé de permettre qu'au moins les flottants puissent être à cheval sur toutes les colonnes, c'est-à-dire, produits par l'environnement figure*. Vous pouvez voir cette nouvelle fonctionnalité dans le tableau 2 qui a été inséré ici même. Cependant, des flottants dans les colonnes uniques sont toujours interdits et je ne pense pas que j'aurai le temps de m'attaquer à ce problème dans un proche avenir. À tous ceux qui veulent essayer : attendez TeX 3.0. Il a quelques caractéristiques qui rendront la vie plus facile dans un environ-

`\setemergencystretch` : C'est un point d'entrée pour les utilisateurs qui aiment jouer finement avec les réglages. Il est supposé fixer le registre `\emergencystretch` ⟨*dimen*⟩ fourni par TEX 3.0. Le premier argument est le nombre de colonnes et le second la `\hsize` courante. Pour l'instant, la définition par défaut est 4 `pt` × `#1`, c'est-à-dire que `\hsize` n'est pas utilisé du tout. Mais, il y a peut-être de meilleures formules.

`\set@floatcmds` : C'est le point d'entrée pour les experts qui aiment implémenter un mécanisme complet de traitement des flottants pour l'environnement multicols. Le `@` dans le nom devrait signaler que ce ne sera pas très facile.

TAB. 2 – Les nouvelles commandes pour multicol.sty, version 1.2. Les deux commandes pourraient être supprimées si de bonnes solutions à ces problèmes non résolus sont trouvées. J'espère que ces commandes empêcheront que des styles semblables dérivés de celui-là ne se promènent dans la nature.

nement multi-colonne. Toutefois, nous travaillons ici à la pointe des possibilités de TEX, et des solutions vraiment parfaites nécessiteraient une approche différente de celle qui était réalisée dans le *page_builder*[9] de TEX.

Le texte ci-dessous est presque inchangé, j'ai seulement ajouté de la documentation aux endroits où le nouveau code a été inséré.

4 L'implémentation

Nous sortons maintenant du mode multi-colonne pour montrer les capacités de cet environnement (et les mauvaises décisions de mise en page)

4.1 Le fichier pilote de documentation

Le petit bout de code suivant contient le fichier pilote de documentation de TEX, *i.e.* le fichier qui produira la documentation que vous êtes en train de lire. Il sera extrait de ce fichier par le programme `docstrip`. Puisque c'est le premier code dans ce fichier, on peut produire la documentation simplement en exécutant LATEX sur le fichier `.dtx`.

```
⟨*driver⟩
\documentclass{ltxdoc}
```

Nous utilisons l'option `balancingshow` quand nous chargeons multicols pour que le traçage complet soit produit. Cela doit être fait avant que le package `doc` ne soit chargé, puisque `doc` exige multicols sans aucune option.

```
\usepackage{multicol}[1999/05/25]
\usepackage{doc}
```

Tout d'abord, nous fixons la mise en page convenable pour cet article.

```
\setlength{\textwidth}{39pc}
\setlength{\textheight}{54pc}
\setlength{\parindent}{1em}
\setlength{\parskip}{0pt plus 1pt}
\setlength{\oddsidemargin}{0pc}
\setlength{\marginparwidth}{0pc}
\setlength{\topmargin}{-2.5pc}
\setlength{\headsep}{20pt}
\setlength{\columnsep}{1.5pc}
```

Nous voulons un filet entre les colonnes.

```
\setlength\columnseprule{.4pt}
```

Nous voulons aussi nous assurer que le nouvel environnement multicols trouve assez de place à la fin de la page.

```
\setlength\premulticols{6\baselineskip}
```

Lors de l'équilibrage des colonnes, nous ne tenons aucun compte des solutions trop mauvaises. De plus, si la dernière colonne est trop mauvaise, nous la composons sans étirement.

```
\setcounter{columnbadness}{7000}
\setcounter{finalcolumnbadness}{7000}
```

L'index est censé apparaître sur quatre colonnes. Et nous ne montrons pas les noms des macros dans la marges.

```
\setcounter{IndexColumns}{4}
\let\DescribeMacro\SpecialUsageIndex
\let\DescribeEnv\SpecialEnvIndex
\renewcommand\PrintMacroName[1]{}
\CodelineIndex
%\DisableCrossrefs            % Partial index
\RecordChanges                % Change log
```

[9] c'est la partie de TEX qui découpe le texte courant en pages, se charge de l'insertion des flottants, des notes, de la composition des en-têtes, ...

Les numéros de lignes sont très petites pour cet article.

```
\renewcommand{\theCodelineNo}
  {\scriptsize\rm\arabic{CodelineNo}}
\settowidth\MacroIndent{\scriptsize\rm 00\ }
⟨/driver⟩
```

```
⟨*driver⟩
\begin{document}
  \typeout
  {***************************************
 ^^J* Expect some Under- and overfull boxes.
 ^^J***************************************}
  \DocInput{f-multicol.dtx}
\end{document}
⟨/driver⟩
```

4.2 Identification et exécution des options

Nous commençons par identifier le package. Puisque son utilisation requiert des caractéristiques seulement disponibles dans $\LaTeX\,2_\varepsilon$, nous nous assurons que ce format est disponible (maintenant, cela est fait plus tôt dans le fichier).

```
⟨*package⟩
% \NeedsTeXFormat{LaTeX2e}
% \ProvidesPackage{multicol}[..../../..
%    v... multicolum formatting]
```

Ensuite, nous déclarons les options prises en charge par multicols. Le mode `twocolumn` et multicols ne fonctionnent pas ensemble, donc nous mettons en garde contre des problèmes possibles. Quoi qu'il en soit, puisque nous pouvons revenir au mode `\onecolumn` dans lequel multicols fonctionne, nous n'entraînons pas d'erreur.

```
\DeclareOption{twocolumn}
   {\PackageWarning{multicol}{May not work
              with the twocolumn option}}
```

Le traçage est fait grace à un compteur. Cependant, il est également possible d'appeler le traçage en utilisant les options déclarées ci-dessous.

```
\newcount\c@tracingmulticols
\DeclareOption{errorshow}
    {\c@tracingmulticols\z@}
\DeclareOption{infoshow}
    {\c@tracingmulticols\@ne}
\DeclareOption{balancingshow}
    {\c@tracingmulticols\tw@}
\DeclareOption{markshow}
    {\c@tracingmulticols\thr@@}
\DeclareOption{debugshow}
    {\c@tracingmulticols5\relax}
\ProcessOptions
```

4.3 Ouvrir et fermer l'environnement multicols

Comme précisé précédemment, l'environnement multicols a un argument obligatoire (le nombre de colonnes) et jusqu'à deux arguments optionnels. Nous commençons par lire le nombre de colonnes dans le registre `\col@number`.

```
\def\multicols#1{\col@number#1\relax
```

Si l'utilisateur oublie l'argument, TEX se plaindra de l'absence d'un nombre à ce point. Le mécanisme de réparation d'erreur utilisera ensuite zéro, qui n'est pas un bon choix dans ce cas. Donc, nous devons maintenant tester si tout est correct. Le minimum est de deux colonnes pour le moment.

```
  \ifnum\col@number<\tw@
     \PackageWarning{multicol}%
      {Using `\number\col@number'
        columns doesn't seem a good idea.^^J
        I therefore use two columns instead}%
     \col@number\tw@ \fi
```

Nous avons seulement assez de registres de boîte pour dix colonnes, donc nous avons besoin de vérifier que l'utilisateur n'en demande pas plus.

```
  \ifnum\col@number>10
     \PackageError{multicol}%
      {Too many columns}%
      {Current implementation doesn't
       support more than 10 columns.%
       \MessageBreak
       I therefore use 10 columns instead}%
     \col@number10 \fi
```

À l'intérieur de l'environnement, nous avons besoin d'une version spéciale de la commande fondamentale `\@footnotetext` puisque l'original définit `\hsize` pour `\columnwidth` ce qui n'est pas correct dans l'environnement multicols. Ici, `\columnwidth` se réfère à la largeur d'une colonne unique et la note de bas de page doit être de longueur `\textwidth`. Puisque `\@footnotetext` a une définition différente dans l'environnement minipage, nous ne le redéfinissons pas directement. À la place, nous fixons localement `\columnwidth` à `\textwidth` et appelons la définition originale (courante) stockée dans `\orig@footnotetext`.

```
     \let\orig@footnotetext\@footnotetext
     \long\def\@footnotetext##1{\begingroup
         \columnwidth\textwidth
```

```
        \orig@footnotetext{##1}\endgroup}%
```

Maintenant, nous pouvons rechercher, en toute sécurité, les arguments optionnels.

```
  \@ifnextchar[\mult@cols{\mult@cols[]}}
```

La macro `\mult@cols` saisit le premier argument optionnel (s'il existe) et recherche le second.

```
\def\mult@cols[#1]{\@ifnextchar[%
```

Cet argument doit être une ⟨*dimen*⟩ indiquant l'espace libre minimum nécessaire sur la page courante pour commencer l'environnement. Si l'utilisateur n'en fournit pas, nous utilisons `\premulticols` par défaut.

```
  {\mult@@cols{#1}}%
  {\mult@@cols{#1}[\premulticols]}}
```

Après avoir enlevé tous les arguments de l'entrée, nous sommes capables de commencer avec `\mult@@cols`.

```
\def\mult@@cols#1[#2]{%
```

La première chose que nous faisons est de décider si, oui ou non, il y a un environnement multicol non lié, *i.e.* à cheval sur plusieurs pages ou composé dans une boîte. Si nous sommes en mode « intérieur » de TeX (par exemple, déjà dans une boîte), alors nous avons une version emboîtée de multicol et par conséquent, nous fixons `@boxedmulticols` à vrai. multicols doit démarrer en mode vertical. Si nous n'y sommes pas déjà, nous le forçons maintenant avec `\par` puisque, autrement, le test de mode « intérieur » ne sera pas fait pas si nous sommes dans une boîte.

```
  \par
  \ifinner \@boxedmulticolstrue
```

Autrement, nous vérifions `\doublecol@number`. Ce compteur est à zéro à l'extérieur de l'environnement multicol mais positif à l'intérieur (cela arrive un petit peu plus tard). Dans le second cas, nous avons besoin de composer les multicolonnes courantes aussi en « mode emboîté » et donc de changer le commutateur en conséquence.

```
  \else
    \ifnum \doublecol@number>\z@
      \@boxedmulticolstrue
    \fi
  \fi
```

Puis nous regardons si des statistiques sont demandées :

```
  \mult@info\z@
     {Starting environment with
      \the\col@number\space columns%
```

En mode emboîté, nous ajoutons quelques info. supplémentaires.

```
       \if@boxedmulticols\MessageBreak
       (boxed mode)\fi
     }%
```

Puis nous mesurons la page courante pour voir si une partie utilisable de l'environnement multicol peut être composée. Cette routine peut v́entuellement commencer une nouvelle page.

```
  \enough@room{#2}%
```

Maintenant nous sortons le premier argument et produisons un espace vertical au-dessus des colonnes (Notez que cet argument correspond au premier argument optionnel de l'environnement multicols). Dans plusieurs versions, cet argument était composé dans un groupe pour donner un effet similaire à celui de `\twocolumn[..]` dans lequel l'argument est aussi implicitement entouré de crochets. Cependant, ceci entre en conflit avec des changements locaux réalisés par des instructions comme les commandes de sectionnement (qui rendent compte de la plupart des commandes utilisées dans cet argument), gâchant l'espacement vertical, etc., plus tard dans le document, de telle sorte qu'à partir de la version v1.5q, cet argument est de nouveau composé dans un niveau extèrieur.

```
  #1\par\addvspace\multicolsep
```

Nous commençons un nouveau niveau de groupement pour cacher tous les changements suivants (réalisés dans `\prepare@multicols` par exemple).

```
  \begingroup
    \prepare@multicols
```

Si nous sommes en mode emboîté, nous ouvrons maintenant une boîte pour y composer tout le matériel à partir du corps multicolonne, sinon nous poursuivons simplement.

```
    \if@boxedmulticols
      \setbox\mult@box\vbox\bgroup
```

Ici, nous avons peut-être à réinitialiser des paramètres ; peut-être `\@parboxrestore` serait la bonne action, mais nous le laissons, pour le moment.

```
    \fi
```

Nous finissons par supprimer les espaces initiaux.

```
    \ignorespaces}
```

Ici est le commutateur et la boîte pour le code des multicolonnes « emboîtées ».

```
\newif\if@boxedmulticols
\@boxedmulticolsfalse
\newbox\mult@box
```

La macro `\enough@room` utilisée ci-dessous n'est pas parfaite mais fonctionne raisonnablement bien dans ce contexte. Nous mesurons l'espace libre sur la page courante en soustrayant `\pagetotal` de `\pagegoal`. Ce n'est pas totalement correct puisque il ne prend pas en compte le « retrécissement » (*i.e.* `\pageshrink`). La « récente liste de contribution » pourrait être non-vide, donc nous commençons avec `\par` et une pénalité implicite [10]. En fait, nous utilisons `\addpenalty` pour nous assurer qu'un `\addvspace` suivant « verra » l'espace vertical qui pourrait être présent. L'utilisation de `\addpenalty` aura pour conséquence que tous les items, provenant des contributions récentes, seront déplacés dans la liste verticale principale et la valeur de `\pagetotal` sera correctement mise à jour. Cependant, la pénalité sera placée devant tout morceau de glue avec pour résultat que la liste verticale principale sera déjà débordante même si TeX n'appelle pas la routine de sortie.

```
\def\enough@room#1{%
```

Mesurer a uniquement un sens quand nous ne sommes pas en « mode emboîté », donc la routine ne fait rien si le commutateur est à vrai.

```
  \if@boxedmulticols\else
  \par
```

Pour vider la liste de contribution, la première version contenait une pénalité zéro, mais cela avait pour conséquence que `\addvspace` ne pouvait détecter la glue précédente. Donc, ceci a été changé en `\addpenalty`. Cela s'est avéré cependant insuffisant puisque `\addpenalty` n'ajoutera pas de pénalité quand `@nobreak` est vrai. Donc, nous forçons localement ce commutateur à faux. Il peut en résulter une rupture entre le texte précédent et le début de l'environnement multicol, mais cela semble acceptable puisqu'il y a un argument optionnel précisément pour cette raison.

```
  \bgroup\@nobreakfalse\addpenalty\z@\egroup
  \page@free \pagegoal
  \advance \page@free -\pagetotal
```

Pour être capable de sortir la valeur, nous avons besoin de l'assigner préalablement à un registre puisqu'elle pourraît être un registre (par défaut), auquel cas il nous faudrait utiliser `\the`, ou elle pourraît être une valeur à part entière et alors utiliser `\the` serait mauvais.

```
    \@tempskipa#1\relax
```

Maintenant nous testons si les informations de traçage sont demandées :

```
  \mult@info\z@
      {Current page:\MessageBreak
       height=%
       \the\pagegoal: used \the\pagetotal
       \space -> free=\the\page@free
       \MessageBreak
       needed \the\@tempskipa
             \space(for #1)}%
```

Notre dernière action est de forcer un saut de page s'il ne reste pas assez de place.

```
   \ifdim \page@free <#1\newpage \fi
  \fi}
```

Lors de la préparation de la sortie multicolonne, plusieurs choses sont à faire.

```
\def\prepare@multicols{%
```

Nous commençons par sauver la valeur courante de `\@totalleftmargin` et puis remettons à zéro `\parshape` dans le cas où nous sommes à l'intérieur de quelque environnement de liste. L'indentation convenable pour l'environnement multicols dans un tel cas sera produite par le déplacement du résultat vers la droite par la commande `\multicol@leftmargin`. Si nous voulions utiliser directement la valeur de `\@totalleftmargin` alors les listes à l'intérieur d'un environnement multicols pourraîent provoquer un décalage de la sortie.

```
  \multicol@leftmargin\@totalleftmargin
  \@totalleftmargin\z@
  \parshape\z@
```

Nous définissons également le registre `\doublecol@number` pour un usage ultérieur. Ce registre doit contenir $2 \times$ `\col@number`. Il indique aussi que nous sommes dans un environnement multicols comme précisé ci-dessus.

```
  \doublecol@number\col@number
  \multiply\doublecol@number\tw@
  \advance\doublecol@number\mult@rightbox

  \if@boxedmulticols
    \let\l@kept@firstmark\kept@firstmark
    \let\l@kept@botmark\kept@botmark
    \global\let\kept@firstmark\@empty
    \global\let\kept@botmark\@empty
  \else
```

Nous ajoutons une boîte vide à la liste verticale principale pour nous assurer que nous prenons toutes les insertions (remises plus tard ou insérées en haut de la page). Sinon, il pourrait arriver que `\eject` soit supprimé sans appel de la routine de sortie. Dans cette routine de sortie, nous enlevons de nouveau cette boîte. De nouveau, le code est appliqué seulement si nous sommes dans la liste verticale principale et non dans une boîte. Cependant, ce n'est pas suffisant pour annuler l'interligne, nous devons aussi effacer `\topskip` avant d'ajouter cette boîte, puisque `\topskip` est toujours inséré avant la première boîte d'une page, ce qui pourrait nous laisser

[10] cf. la documentation de `\endmulticols` pour plus de détails

avec un espace supplémentaire de `\topskip` si multicols commence sur une nouvelle feuille.

```
    \nointerlineskip {\topskip\z@\null}%
    \output{%
      \global\setbox\partial@page\vbox
        {%
```

Maintenant, nous devons nous assurer que nous prenons en compte une situation particulière qui pourrait causer de la perte de texte ! Si l'utilisateur a une énorme quantité de matériel vertical dans le premier argument optionnel, qui est plus grand que `\premulticols` et que nous sommes près du bas de page, alors il peut arriver que ce ne soit pas `\eject` qui déclenche cette routine spéciale de sortie, mais plutôt la liste verticale principale débordante. Dans ce cas, nous donnons un autre point de rupture à travers la pénalité de `\eject`. Par conséquent, cette routine spéciale de sortie peut être appelée deux fois et le contenu de `\partial@page`, *i.e.* le matériel avant l'environnement multicols, est perdu. Il y a plusieurs solutions pour contourner ce problème, mais pour l'instant, nous le détecterons simplement et informerons l'utilisateur qu'il/elle doit élargir `\premulticols` en utilisant une valeur appropriée pour le second argument.

```
⟨*check⟩
         \ifvoid\partial@page\else
           \PackageError{multicol}%
            {Error saving partial page}%
            {The part of the page before
             the multicols environment was
             nearly full with^^Jthe result
             that starting the environment
             will produce an overfull
             page.  Some^^Jtext may be lost!
             Please increase \premulticols
             either generally or for this%
             ^^Jenvironment by specifying a
             suitable value in the second
             optional argument to^^Jthe
             multicols environment.}
           \unvbox\partial@page
           \box\last@line
         \fi
⟨/check⟩
         \unvbox\@cclv
         \global\setbox\last@line\lastbox
        }%
```

Finalement, nous avons besoin d'enregistrer les balises qui sont toujours présentes à l'intérieur de `\partial@page` afin de pouvoir construire plus tard des balises correctes de début et de fin. Cela est réalisé par le code suivant.

```
           \prep@keptmarks
```

Finalement, nous devons initialiser `\kept@topmark` qui, dans l'idéal, devrait être initialisé avec la balise qui est en cours en « haut » de la page. Malheureusement, nous ne pouvons utiliser `\topmark` parce que ce registre ne contiendra pas toujours ce que son nom veut dire, parce que, parfois, LaTeX appelle la routine de sortie pour la gestion des flottants [11]. Donc, nous utilisons le deuxième meilleure solution, l'initialisant avec `\firstmark`. En fait, dans notre optique, cela n'a vraiment pas d'importance, puisque nous utilisons `\kept@topmark` uniquement pour initialiser `\firstmark` et `\botmark` pour la page suivante, si nous n'avons pas trouvé de balise pour la page en cours.

```
           \global\let\kept@topmark\firstmark
          }\eject
```

La chose suivante à faire est d'assigner une nouvelle valeur à `\vsize`. LaTeX conserve la place libre de la page (*i.e.* la hauteur de page sans l'espace pour les flottants déjà pris en compte) dans le registre `\@colroom`. Nous devons soustraire la hauteur de `\partial@page` pour mettre la véritable place libre dans cette variable.

```
    \advance\@colroom-\ht\partial@page
```

Puis, nous devons calculer la valeur de `\vsize` à utiliser pendant l'assemblage des colonnes. `\set@mult@vsize` prend un argument qui permet les définitions locales (`\relax`) ou globales (`\global`). La dernière possibilité est utilisée à l'intérieur de la routine de sortie ci-dessous. À cet instant, nous devons réaliser un changement local pour `\vsize` parce que nous voulons restaurer la valeur originale de `\vsize` dans le cas où l'environnement multicols finit sur la même page que celle sur laquelle il a commencé.

```
    \set@mult@vsize\relax
```

Maintenant nous passons à une nouvelle routine `\output` qui sera utilisée pour rassembler le contenu des colonnes.

```
    \output{\multi@column@out}%
```

Finalement, nous gérons les insertions des notes de bas de page. Nous devons multiplier le facteur d'agrandissement et le saut supplémentaire par le nombre de colonnes puisque chaque note de bas de page réduit l'espace de chaque colonne (rappelez-vous que les notes de bas de page s'étendent sur toute la largeur de la page). Si, d'autre part, les notes sont composées tout à la fin du document, notre schéma fonctionnera encore, puisque

[11] pendant un tel appel, `\botmark` est copié globalement dans `\topmark` par le programme TeX

`\count\footins` est égal à zéro et, donc, il ne changera pas. Pour permettre une personnalisation avancée, la définition des paramètres `\footins` est réalisée dans une macro séparée.

```
    \init@mult@footins
```

Pour la même raison (notes de la largeur de la page), le registre ⟨*dimen*⟩ contrôlant l'espace maximum utilisé pour les notes de bas de page n'est pas modifié. Ayant fait cela, nous devons ré-insérer toutes les notes de bas de page qui sont déjà présentes (*i.e.* celles rencontrées quand le matériel sauvé dans `\partial@page` a été traité en premier). Cela réduira l'espace libre (*i.e.* `\pagetotal`) de la quantité appropriée puisque nous avons changé le facteur d'agrandissement, etc., ci-dessus.

```
    \reinsert@footnotes
```

Tout le code ci-dessus était seulement nécessaire pour la version de multicols non-restreinte, *i.e.* celle qui permet les ruptures de page. Si nous sommes à l'intérieur d'une boîte, il n'y a pas de raison de définir les routines spéciales de sortie, ou `\vsize`, etc.

```
  \fi
```

Mais maintenant, nous en venons au code qui est nécessaire dans tous les cas. Nous assignons de nouvelles valeurs à `\vbadness`, `\hbadness` et `\tolerance` puisque il est assez dur pour TEX de produire de « bons » paragraphes à l'intérieur de colonnes étroites.

```
  \vbadness\@Mi \hbadness5000
  \tolerance\multicoltolerance
```

Puisque la première passe échouera presque toujours, nous l'ignorons et demandons à TEX de réaliser les césures directement. En fait, maintenant nous utilisons un autre registre pour garder la valeur pour la pré-tolérance de multicol, afin qu'un concepteur puisse utiliser `\pretolerance`.

```
  \pretolerance\multicolpretolerance
```

Pour l'utilisation avec le nouveau TEX, nous fixons `\emergencystretch` à `\col@number`× 4 pt. Cependant, c'est seulement une estimation, donc pour l'instant cela est réalisé dans une macro, `\setemergencystretch` qui prend la valeur courante de `\hsize` et le nombre de colonnes comme argument. Donc, les utilisateurs sont capables de construire leur propre formule.

```
  \setemergencystretch\col@number\hsize
```

Un autre point d'entrée, pour permettre aux utilisateurs d'ajouter leurs propres extensions sans faire un nouveau package, est `\set@floatcmds` qui gèrent toutes les redéfinitions des commandes de flottant internes à LATEX pour fonctionner avec l'environnement multicols. Pour le moment, c'est seulement utilisé pour redéfinir `\@dblfloat` et `\end@dblfloat`.

```
  \set@floatcmds
```

De plus, nous augmentons `\baselineskip` de `\multicolbaselineskip` pour permettre les corrections des colonnes étroites.

```
  \advance\baselineskip\multicolbaselineskip
```

La valeur `\hsize` des colonnes est donnée par la formule :

$$\frac{\texttt{\textbackslash linewidth} - (\texttt{\textbackslash col@number} - 1) \times \texttt{\textbackslash columnsep}}{\texttt{\textbackslash col@number}}$$

La formule ci-dessus a changé de version en version. Maintenant, nous commençons avec la valeur courante de `\linewidth` afin que la largeur de la colonne soit correctement calculée quand nous sommes dans une minipage, dans une liste ou dans tout autre environnement. Cela sera accompli par :

```
  \hsize\linewidth \advance\hsize\columnsep
  \advance\hsize-\col@number\columnsep
  \divide\hsize\col@number
```

Nous fixons également `\linewidth` et `\columnwidth` à `\hsize`. Dans le passé, `\columnwidth` avait été laissé tel quel. C'est incohérent mais `\columnwidth` est utilisé seulement par les flottants (qui ne sont pas permis dans leur implémentation courante) et par la macro `\footnote`. Puisque nous voulons des notes de bas de page de la largeur de la page[12], cette simple astuce nous évite de ré-écrire les macros `\footnote`. Cependant, certaines applications se réfèrent à `\columnwidth` comme « la largeur de la colonne courante » pour composer les affichages (le package `amsmath`, par exemple), et pour permettre l'utilisation de telles applications en même temps que `multicol`, cela a été maintenant modifié.

Avant de remplacer `\linewidth` par sa nouvelle valeur, nous enregistrons son ancienne valeur dans un registre appelé `\full@width`. Cette valeur est utilisée plus tard, quand nous rassemblons toutes les colonnes.

```
  \full@width\linewidth
  \linewidth\hsize
  \columnwidth\hsize
}
```

[12] Je ne suis pas sûr de vouloir des notes de la largeur de la page, mais l'équilibrage de la dernière page ne peut être accompli qu'avec cette approche ou avec un algorithme « multi-path » qui est compliqué et lent. C'est toutefois un défi pour qui veut me démontrer que c'est mauvais! Une autre possibilité est de réimplémenter une petite partie de la procédure *fire_ up* de TEX (le programme). Je pense que c'est la meilleure solution si vous êtes intéressés par la composition complexe de pages, mais cela a pour inconvénient que le programme résultant ne peut plus être appelé TEX.

Cette macro est utilisée pour fixer les paramètres associés avec les notes (de bas de page) flottantes. Elle peut être redéfinie par des applications qui demandent des quantités d'espace différentes lors de la composition des notes.

```
\def\init@mult@footins{%
    \multiply\count\footins\col@number
    \multiply\skip \footins\col@number
}
```

Puisque nous avons fixé les colonnes `\col@umber` sur une page, chacune avec une hauteur de `\@colroom`, nous devons assigner `\vsize` = `\col@number` × `\@colroom` pour rassembler assez de matériel avant d'entrer de nouveau dans la routine `\output`. En fait, nous devons encore ajouter (`\col@number`−1)× (`\baselineskip` − `\topskip`) si vous y réfléchissez sérieusement.

```
\def\set@mult@vsize#1{%
    \vsize\@colroom
    \@tempdima\baselineskip
    \advance\@tempdima-\topskip
    \advance\vsize\@tempdima
    \vsize\col@number\vsize
    \advance\vsize-\@tempdima
```

Mais cela pourrait être insuffisant puisque nous utilisons plus tard `\vsplit` pour extraire les colonnes du matériel réuni. C'est pourquoi nous ajoutons des « lignes supplémentaires », leur nombre dépendant de la valeur du compteur « multicols ». La valeur finale est assignée globalement si `#1` est `\global` parce ce que nous voulons utiliser cette macro, plus tard, aussi à l'intérieur de la routine de sortie.

```
    #1\advance\vsize
        \c@collectmore\baselineskip}
```

Voici le registre ⟨*dimen*⟩ dont nous avons besoin pour sauvegarder la valeur externe de `\@totalleftmargin`.

```
\newdimen\multicol@leftmargin
```

Quand la fin de l'environnement multicols est détectée, nous devons équilibrer le matériel réuni. Que nous soyons ou non à l'intérieur d'une multicol emboîtée, différentes choses peuvent survenir. Mais, d'abord, nous terminons le paragraphe courant avec une commande `\par`.

```
\def\endmulticols{\par
  \if@boxedmulticols
```

En mode emboîté, nous devons fermer la boîte dans laquelle nous avons rassemblé tout le matériel pour les colonnes.

```
    \egroup
```

Maintenant, nous appelons `\balance@columns`, la routine qui équilibre le matériel stocké dans la boîte `\mult@box`.

```
    \balance@columns
```

Après équilibrage, le résultat doit être retourné par la commande `\page@sofar`. Mais avant de le faire, nous réinsérons toutes les balises trouvées dans la boîte `\mult@box`.

```
    \return@nonemptymark{first}%
                  \kept@firstmark
    \return@nonemptymark{bot}%
                  \kept@botmark
    \page@sofar

    \global\let\kept@firstmark
                \l@kept@firstmark
    \global\let\kept@botmark
            \l@kept@botmark
⟨*marktrace⟩
    \mult@info\tw@
      {Restore kept marks to\MessageBreak
       first: \meaning\kept@firstmark
       \MessageBreak bot\space\space:
                      \meaning\kept@botmark }%
⟨/marktrace⟩
```

Cela finit le code pour les cas « emboîtés ».

```
  \else
```

Si nous sommes dans un environnement multicols non-restreint, nous fermons le paragraphe courant par `\par`, mais ce n'est pas suffisant puisque le *page_builder* TEX ne videra pas totalement la liste de contribution [13]. Donc, nous devons aussi ajouter un `\penalty` explicite. La liste de contribution sera désormais vidée et, si son matériel ne tient pas entièrement dans la page courante, alors la routine de sortie sera appelée avant que nous la changions. À ce moment, nous avons besoin d'utiliser `\penalty` et non `\addpenalty` pour nous assurer que a) les récentes contributions sont vidées et b) le tout dernier item de la liste verticale principale est un point de rupture valide et donc que TEX coupera la page dans le cas où elle est débordante.

```
    \penalty\z@
```

Maintenant, il n'est pas dangereux de modifier la routine de sortie pour équilibrer les colonnes.

```
    \output{\balance@columns@out}\eject
```

[13] Cela causait auparavant un bug fort intriguant dans lequel du matériel était équilibré deux fois, ce qui causait des superpositions. La raison était que `\eject` était placé à la fin de la liste de contribution. Le *page_builder* était alors appelé (un `\penalty` explicite videra la liste de contribution), mais la ligne avec `\eject` ne tenait pas sur la page courante. Il était ensuite repris en compte aprés la fin de la routine de sortie, entraînant une seconde rupture après une ligne

Si le corps de l'environnement multicols est complètement vide, ou si un multicols multi-page finit juste à une liaison entre page, nous avons le cas inhabituel où `\eject` n'aura aucun effet (puisque la liste verticale principale est vide) – donc, aucune routine de sortie ne sera jamais appelée. En conséquence, le matériel précédant multicols (stocké dans `\partial@page`) sera perdu si on ne gère pas ça à la main.

```
      \ifvbox\partial@page
          \unvbox\partial@page\fi
```

Après l'action de la routine de sortie, nous restaurons les balises sauvegardées à leur valeur initiale.

```
      \global\let\kept@firstmark\@empty
      \global\let\kept@botmark\@empty
⟨*marktrace⟩
      \mult@info\tw@
        {Make kept marks empty}%
⟨/marktrace⟩
   \fi
```

La routine de sortie ci-dessus s'occupera de `\vsize` et réinsèrera les colonnes équilibrées, etc. Mais elle ne pourra réinsérer `\footnotes` parce que nous devons d'abord restaurer le paramètre `\footins`, puisque nous sommes revenus en mode une colonne. Cela sera réalisé dans la prochaine ligne de code ; nous fermons simplement le groupe commencé dans `\multicols`.

Pour corriger un sombre bug qui est le résultat de la définition courante des macros `\begin` ... `\end`, nous vérifions que nous sommes encore (logiquement parlant) dans l'environnement multicols. Si, par exemple, nous oublions de fermer quelque environnement à l'intérieur de l'environnement multicols, le `\endgroup` suivant devrait être considéré de façon incorrecte pour la fermeture de cet environnement.

```
   \@checkend{multicols}%
   \endgroup
```

Maintenant, il est temps de retourner toutes les notes de bas de page, si nous sommes en mode non-restreint :

```
   \if@boxedmulticols\else
     \reinsert@footnotes
   \fi
```

Nous fixons également le compteur « unbalance » à sa valeur par défaut. Cela est réalisé globalement puisque les compteurs LaTeX sont toujours modifiés de cette manière [14].

```
   \global\c@unbalance\z@
```

Nous jetons aussi un coup d'œil à la quantité d'espace libre dans la page courante, pour voir s'il est temps de faire un saut de page. L'espace vertical ajouté par la suite disparaîtra si `\enough@room` commence une nouvelle page.

```
   \enough@room\postmulticols
   \addvspace\multicolsep
```

Si des statistiques sont demandées, nous annonçons finalement que nous avons tout terminé.

```
   \mult@info\z@
      {Ending environment
              \if@boxedmulticols
                 \space(boxed mode)\fi
             }}
```

Finissons cette section en allouant tous les registres déjà utilisés.

```
\newcount\c@unbalance
\newcount\c@collectmore
```

Dans la nouvelle version de LaTeX, `\col@number` est déjà alloué par le noyau, donc nous ne l'allouons pas de nouveau.

```
%\newcount\col@number
\newcount\doublecol@number
\newcount\multicoltolerance
\newcount\multicolpretolerance
\newdimen\full@width
\newdimen\page@free
\newdimen\premulticols
\newdimen\postmulticols
\newskip\multicolsep
\newskip\multicolbaselineskip
\newbox\partial@page
\newbox\last@line
```

Et voilà leurs valeurs par défaut :

```
\c@unbalance   = 0
\c@collectmore = 0
```

Pour permettre de vérifier si quelque macro est utilisée à l'intérieur de l'environnement multicols, le compteur `\col@number` prend par défaut la valeur `1` à l'extérieur de l'environnement.

```
\col@number = 1
\multicoltolerance = 9999
\multicolpretolerance = -1
\premulticols = 50pt
\postmulticols= 20pt
\multicolsep = 12pt plus 4pt minus 3pt
\multicolbaselineskip=0pt
```

[14] En fait, nous sommes toujours dans un groupe commencé par la macro `\begin` et donc `\global` doit être utilisé de toute façon

4.4 Les routines de sortie

Tout d'abord, nous commençons par quelques macros simples. Lors de la composition de la page, nous sauvegardons les colonnes soit dans les registres de boîte 0, 2, 4,... (localement) soit 1, 3, 5,... (globalement). C'est la règle de PLAIN TEX pour éviter un débordement des piles stockées.

Donc, nous définissons une macro `\process@cols` pour nous aider à utiliser ces registres dans les routines de sortie ci-dessous. Elle a deux arguments : le premier est un nombre ; le second est l'information d'exécution. Elle commence la boucle avec `\count@=#1` (`\count@` est un registre temporaire défini en PLAIN TEX), exécute l'argument `#2`, ajoute deux à `\count@`, exécute de nouveau l'argument `#2`, etc. jusqu'à ce que `\count@` soit plus grand que `\doublecol@number`. Il pourrait être plus facile de comprendre grâce à un exemple, donc nous la définissons maintenant et expliquons son utilisation plus tard.

```
\def\process@cols#1#2{\count@#1\relax
      \loop
⟨*debug⟩
      \typeout{Looking at box \the\count@}
⟨/debug⟩
       #2%
      \advance\count@\tw@
      \ifnum\count@<\doublecol@number
   \repeat}
```

Nous définissons maintenant `\page@sofar` pour donner un exemple de la macro `\process@cols`. `\page@sofar` devrait sortir tout ce qui est préparé par la routine d'équilibrage, `\balance@columns`.

```
\def\page@sofar{%
```

`\balance@columns` prépare sa sortie dans les registres temporaires de boîte numérotés pairs. Maintenant, nous sortons les colonnes réunies en supposant qu'elles sont sauvées dans les registres de boîte 2 (colonne de gauche), 4 (seconde colonne) ... Cependant, la dernière colonne (*i.e.* la plus à droite) devrait être sauvée dans le registre de boîte 0[15]. D'abord, nous nous assurons que les colonnes ont des largeurs identiques. Nous utilisons `\process@cols` dans ce but, commençant avec `\count@` = `\mult@rightbox`. Donc `\count@` boucle suivant `\mult@rightbox`, `\mult@rightbox` + 2,... (jusqu'à `\doublecol@number`).

```
    \process@cols\mult@rightbox
```

Nous devons vérifier si la boîte en question est vide, parce que l'opération `\wd`⟨*number*⟩ sur une boîte vide *ne* changera *pas* ses dimensions (*soupir*).

```
        {\ifvoid\count@
           \setbox\count@\hbox to\hsize{}%
         \else
           \wd\count@\hsize
         \fi}%
```

Maintenant, nous donnons des informations de traçage.

```
    \mult@info\z@
     {Column spec:\MessageBreak
        (\the\multicol@leftmargin\space -->
         \the\full@width\space = \the\hsize
         \space x \the\col@number)%
     }%
```

Pour le moment, nous devons toujours être en mode vertical.

```
\ifvmode\else\errmessage{Multicol Error}\fi
```

Maintenant, nous mettons ensemble toutes les colonnes dans `\hbox` de largeur `\full@width` (la modifiant par `\multicol@leftmargin` à droite afin qu'elle soit placée correctement si nous sommes dans un environnement de liste).

```
    \moveright\multicol@leftmargin
     \hbox to\full@width{%
```

et séparons les colonnes par un filet, si on le désire.

```
      \process@cols\mult@gfirstbox{\box\count@
        \hss\vrule\@width\columnseprule\hss}%
```

Comme vous le remarquerez, nous commençons avec le registre de boîte `\mult@gfirstbox` (*i.e.* la colonne de gauche). Alors, cette fois, `\count@` boucle à travers 2, 4, ... (plus le décalage approprié). Finalement, nous ajoutons la boîte 0 et fermons `\hbox`.

```
      \box\mult@rightbox
```

Les profondeurs des colonnes dépendent de leurs dernières lignes. Pour nous assurer que nous aurons toujours la même apparence en ce qui concerne les filets, nous forçons la profondeur à être au moins celle de la lettre « p ».

```
%     \strut
      \rlap{\phantom p}%
}%
}
```

Avant d'attaquer les plus grosses routines de sortie, nous définissons juste une macro de plus qui nous aidera à trouver notre chemin à travers les mystères plus tard. `\reinsert@footnotes` fera ce que son nom indique : elle réinsère les notes de bas de

[15] Vous verrez la raison ce cette numérotation, quand nous regarderons les routines de sortie `\multi@column@out` et `\balance@columns@out`.

page présentes dans `\footinbox` et donc elles seront ré-exécutées par le *page_builder* de TEX.

Au lieu de vraiment réinsérer les notes de bas de page, nous insérons une note vide. Cela déclenchera le mécanisme d'insertion et, puisque les anciennes notes sont encore dans leur boîte et que nous sommes sur une nouvelle page, `\skip footins` sera correctement pris en compte.

```
\def\reinsert@footnotes{\ifvoid\footins\else
          \insert\footins{}\fi}
```

Maintenant, nous ne pouvons plus repousser les difficultés plus loin. La routine `\multi@column@out` sera appelée dans deux situations. Soit la page est pleine (*i.e.* nous avons accumulé assez de matériel pour construire toutes les colonnes désirées) ou un flottant ou la note de marge (ou bien encore un `\clearpage`) est détecté. Dans le dernier cas, `\outputpenalty` est inférieur à -10000, sinon la pénalité qui déclenche la routine de sortie est plus grande. Donc, il est facile de faire la distinction entre les deux cas : nous testons simplement ce registre.

```
\def\multi@column@out{%
   \ifnum\outputpenalty <-\@M
```

Si c'était un `\clearpage`, un flottant ou une note de marge, nous appelons `\speci@ls`.

```
   \speci@ls \else
```

sinon, nous construisons la page définitive.

Pour la partie du code qui suit, voir les commentaires dans la section 7.2.

```
⟨*colbreak⟩
   \ifvoid\colbreak@box\else
     \mult@info\@ne{Re-adding forced
               break(s) for splitting}%
     \setbox\@cclv\vbox{%
        \unvbox\colbreak@box
        \penalty-\@Mv\unvbox\@cclv}%
   \fi
⟨/colbreak⟩
```

Considérons le cas normal. Nous avons à `\vsplit`er les colonnes à partir du matériel accumulé dans la boîte 255. Donc, nous assignons d'abord les valeurs appropriées à `\splittopskip` et `\splitmaxdepth`.

```
   \splittopskip\topskip
   \splitmaxdepth\maxdepth
```

Puis nous calculons la hauteur de la colonne courante (dans `\dimen@`). Notez que la hauteur de `\partial@page` est déjà soustraite de `\@colroom`, donc, nous pouvons utiliser sa valeur comme point de départ.

```
   \dimen@\@colroom
```

Mais nous devons aussi soustraire l'espace occupé par les notes de bas de page sur la page courante. Notez que, d'abord, nous avons à remettre le registre de saut à sa valeur initiale. Encore une fois, la véritable action est de transposer dans une macro utilitaire, afin que d'autres applications puissent la modifier.

```
   \divide\skip\footins\col@number
   \ifvoid\footins \else
      \leave@mult@footins
   \fi
```

Maintenant, nous sommes capables de `\vsplit`er toutes les colonnes, sauf la dernière. Rappelez-vous que ces colonnes doivent être sauvées dans les registres de boîte 2, 4, ... (plus le décalage).

```
   \process@cols\mult@gfirstbox{%
        \setbox\count@
            \vsplit\@cclv to\dimen@
```

Après séparation, nous mettons à jour les balises conservées.

```
            \set@keptmarks
```

Si `\raggedcolumns` est forcé, nous ajoutons `vfill` au bas en désemboîtant la boîte séparée.

```
            \ifshr@nking
              \setbox\count@
                 \vbox to\dimen@
                  {\unvbox\count@\vfill}%
            \fi
           }%
```

Puis la dernière colonne suit.

```
   \setbox\mult@rightbox
       \vsplit\@cclv to\dimen@
   \set@keptmarks
   \ifshr@nking
      \setbox\mult@rightbox\vbox to\dimen@
          {\unvbox\mult@rightbox\vfill}%
   \fi
```

Ayant fait cela, nous espérons que la boîte 255 est vidée. Si elle ne l'est pas, nous réinsérons son contenu.

```
   \ifvoid\@cclv \else
       \unvbox\@cclv
       \penalty\outputpenalty
```

Dans ce cas, une note de bas de page, qui peut être insérée dans le petit espace restant, sera composée sur la mauvaise page. Donc, nous avertissons l'utilisateur si la page courante contient des notes. Les anciennes versions de multicols produisaient cet avertissement qu'il y ait ou non des notes, ce qui entraînait beaucoup d'avertissements inutiles.

```
       \ifvoid\footins\else
         \PackageWarning{multicol}%
          {I moved some lines to
           the next page.\MessageBreak
           Footnotes on page
           \thepage\space might be wrong}%
       \fi
```

Si le compteur « tracingmulticols » est à 4 ou plus, nous ajoutons également un filet.

```
      \ifnum \c@tracingmulticols>\thr@@
                     \hrule\allowbreak \fi
  \fi
```

Pour donner des balises correctes pour la page courante, nous devons ředéfinir (localement) `\firstmark` and `\botmark`. Si `\kept@firstmark` n'est pas vide, alors `\kept@botmark` ne doit pas être vide non plus, donc nous pouvons utiliser leurs valeurs, Sinon, nous utilisons la valeur de `\kept@topmark` qui a d'abord été initialisé quand nous avons réuni `\partical@page` et plus tard, a été mise à jour au `\botmark` pour la page précédente.

```
  \ifx\@empty\kept@firstmark
     \let\firstmark\kept@topmark
     \let\botmark\kept@topmark
  \else
     \let\firstmark\kept@firstmark
     \let\botmark\kept@botmark
  \fi
```

Nous initialisons également `\topmark` avec `\kept@topmark`. Cela rendra cette balise correcte pour toutes les pages médianes de l'environnement multicols.

```
  \let\topmark\kept@topmark
⟨*marktrace⟩
  \mult@info\tw@
       {Use kept top mark:\MessageBreak
         \meaning\kept@topmark
        \MessageBreak
        Use kept first mark:\MessageBreak
         \meaning\kept@firstmark
       \MessageBreak
        Use kept bot mark:\MessageBreak
         \meaning\kept@botmark
       \MessageBreak
        Produce first mark:\MessageBreak
         \meaning\firstmark
       \MessageBreak
       Produce bot mark:\MessageBreak
         \meaning\botmark
        \@gobbletwo}%
⟨/marktrace⟩
```

Avec un petit peu plus d'effort, nous aurions pu faire mieux. Si nous avions, par exemple, enregistré le rétrécissement du matériel dans `\partial@page`, il serait maintenant possible d'essayer les valeurs supérieures pour `\dimen@` (*i.e.* la hauteur des colonnes) pour passer outre le problème de la boîte 255 non vide. Mais cela aurait rendu le code beaucoup plus complexe, donc je l'ai ignoré (le problème) dans l'implémentation actuelle.

Maintenant, nous utilisons le mécanisme standard de sortie de LaTeX [16]. C'est vrai que c'est une façon de faire amusante.

```
  \setbox\@cclv\vbox{\unvbox\partial@page
                     \page@sofar}%
```

La macro `\@makecol` ajoute à la page courante tous les flottants assignés à cette page. `\@outputpage` débarque la boîte résultante. Notez qu'il est seulement possible que de tels flottants soient présents même si nous n'en permettons aucun dans un environnement multicols.

```
  \@makecol\@outputpage
```

Aprés la sortie de la page, nous devons préparer les marques conservées pour la page suivante. `\kept@firstmark` et `\kept@botmark` sont réinitialisées en les fixant à `\@empty`. La valeur de `\botmark` est ensuite assignée à `\kept@topmark`.

```
    \global\let\kept@topmark\botmark
    \global\let\kept@firstmark\@empty
    \global\let\kept@botmark\@empty
⟨*marktrace⟩
    \mult@info\tw@
       {(Re)Init top mark:\MessageBreak
        \meaning\kept@topmark
        \@gobbletwo}%
⟨/marktrace⟩
```

Maintenant nous fixons `\@colroom` à `\@colht`, qui est la valeur LaTeX sauvegardée pour `\textheight`.

```
  \global\@colroom\@colht
```

Puis, nous traitons les flottants en suspens attendant leur tour pour être placés sur la page suivante.

```
  \process@deferreds
  \@whilesw\if@fcolmade\fi{\@outputpage
     \global\@colroom\@colht
     \process@deferreds}%
```

Si l'utilisateur demande des statistiques, nous l'informons sur la quantité d'espace réservée aux flottants.

```
  \mult@info\@ne
    {Colroom:\MessageBreak
     \the\@colht\space
             after float space removed
             = \the\@colroom \@gobble}%
```

Ayant fait tout cela, nous devons préparer la prise en compte de la page suivante. Donc, nous assignons une nouvelle valeur à `\vsize`. Nouvelle, parce que `\partial@page` est maintenant vide et que `\@colroom` pourrait être diminué de l'espace réservé aux flottants.

```
   \set@mult@vsize \global
```

[16] Cela est extrêmement coûteux car les deux routines de sortie sont gardées en mémoire. La bonne solution aurait été de refaire entièrement la routine de sortie utilisée dans LaTeX

Le registre de saut `\footins` sera ajusté quand le groupe de sortie sera fermé.

```
  \fi}
```

Cette macro est utilisée pour soustraire la quantité d'espace, occupée par les notes de bas de page, de l'espace courant disponible pour la colonne courante. L'espace de la colonne courante est stocké dans `\dimen@`. Voir ci-dessus la description de l'action par défaut.

```
\def\leave@mult@footins{%
    \advance\dimen@-\skip\footins
    \advance\dimen@-\ht\footins
}
```

Nous avons laissé de côté deux macros : `\process@deferreds` et `\speci@ls`.

```
\def\speci@ls{%
 \ifnum\outputpenalty <-\@Mi
```

Si le document finit au milieu d'un environnement multicolonne, par exemple si l'utilisateur a oublié `\end{multicols}`, TEX ajoute une pénalité très négative àu texte en cours de traitement qui est censée signaler à la routine de sortie qu'il est temps de se préparer à expédier tout ce qui reste. Puisqu'à l'intérieur de multicol, la routine de sortie de LATEX est quelquefois désactivée, il est mieux pour nous de vérifier ce cas : si nous trouvons une pénalité très négative, nous produisons un message d'erreur et exécutons la routine de sortie standard dans ce cas.

```
  \ifnum \outputpenalty<-\@MM
   \PackageError{multicol}{Document end
             inside multicols environment}\@ehd
   \@specialoutput
  \else
```

Pour la prochaine partie du code, voir les commentaires dans la section 7.2.

```
⟨*colbreak⟩
      \ifnum\outputpenalty = -\@Mv
          \mult@info\@ne{Forced column
                         break seen}%
          \global\advance\vsize-\pagetotal
          \global\setbox\colbreak@box
                 \vbox{\ifvoid\colbreak@box
                       \else
                         \unvbox\colbreak@box
                         \penalty-\@Mv
                       \fi
                       \unvbox\@cclv}
          \reinsert@footnotes
      \else
⟨/colbreak⟩
```

Si nous rencontrons un flottant ou une note de marge dans l'implémentation actuelle, nous avertissons simplement l'utilisateur que cela n'est pas permis. Puis, nous réinsérons la page et ses notes de bas de page.

```
          \PackageWarning{multicol}%
             {Floats and marginpars not
              allowed inside `multicols'
              environment!
             \@gobble}%
          \unvbox\@cclv\reinsert@footnotes
```

De plus, nous vidons `\@currlist` pour éviter des messages d'erreur futurs quand la routine de sortie de LATEX sera forcée de nouveau. Mais d'abord, nous devons replacer les boîtes dans `\@freelist` (la valeur par défaut de `\@elt` est `\relax` donc cela est possible avec `\xdef`).

```
         \xdef\@freelist{\@freelist\@currlist}%
         \gdef\@currlist{}%
⟨*colbreak⟩
      \fi
⟨/colbreak⟩
  \fi
```

Si la pénalité est -1000, elle proviendra de `\clearpage` et nous exécuterons `\@doclearpage` pour nous débarrasser de tous les flottants en attente.

```
 \else \@doclearpage \fi
}
```

`\process@deferreds` est une version simplifiée de la commande `\@startpage` de LATEX. Nous appelons d'abord `\@floatplacement` pour sauvegarder les paramètres utilisateurs courants dans les registres internes. Puis nous commençons un nouveau groupe et sauvegardons temporairement `\@deferlist` dans la macro `\@tempb`.

```
\def\process@deferreds{%
   \@floatplacement
   \@tryfcolumn\@deferlist
   \if@fcolmade\else
     \begingroup
    \let\@tempb\@deferlist
```

Notre prochaine action est de vider (globalement) `\@deferlist` et assigner une nouvelle signification à `\@elt`. Ici, `\@scolelt` est une macro qui regarde les boîtes dans une liste pour décider si elles doivent être placées sur une nouvelle page (*i.e.* dans `\@toplist` ou `\@botlist`) ou si elles doivent attendre un autre traitement.

```
        \gdef\@deferlist{}%
        \let\@elt\@scolelt
```

Maintenant, nous appelons `\@tempb` qui est de la forme

`\@elt`⟨*box register*⟩`\@elt`⟨*box register*⟩...

Donc `\@elt` (*i.e.* `\@scolelt`) distribuera les boîtes aux trois listes.

```
        \@tempb \endgroup
    \fi}
```

Les déclarations de `\raggedcolumns` et `\flushcolumns` sont définies à l'aide d'une nouvelle macro `\if...`.

```
\newif\ifshr@nking
```

Les définitions réelles sont simples : nous commutons juste à `true` ou `false` en fonction de l'action désirée. Pour éviter des espaces supplémentaires en sortie, nous encapsulons ces changements dans `\@bsphack`...`\@esphack`.

```
\def\raggedcolumns{%
    \@bsphack\shr@nkingtrue\@esphack}
\def\flushcolumns{%
    \@bsphack\shr@nkingfalse\@esphack}
```

Maintenant, la dernière partie du spectacle : la routine de sortie d'équilibrage des colonnes. Puisque ce code est appelé par une pénalité explicite (`\eject`), il n'est pas besoin de vérifier des trucs spéciaux (par exemple, les flottants). Nous commençons par équilibrer le matériel réuni.

```
\def\balance@columns@out{%
```

Pour cela, nous avons besoin de mettre de contenu de la boîte 255 dans `\mult@box`.

```
⟨−colbreak⟩ \setbox\mult@box
⟨−colbreak⟩    \vbox{\unvbox\@cclv}%
```

Pour la partie suivante du code, voir les commentaires dans la section 7.2.

```
⟨*colbreak⟩
    \setbox\mult@box\vbox{%
        \ifvoid\colbreak@box\else
          \unvbox\colbreak@box\break
          \mult@info\@ne{Re-adding
            forced break(s) in balancing}%
        \fi
        \unvbox\@cclv}%
⟨/colbreak⟩
    \balance@columns
```

Cela nous amènera dans la position dans laquelle nous appliquerons `\page@sofar`. Mais d'abord, nous devons fixer `\vsize` à une valeur idoine pour la sortie d'une colonne.

```
    \global\vsize\@colroom
    \global\advance\vsize\ht\partial@page
```

Puis nous `\unvbox`ons `\partial@page` (qui peut être vide si nous n'avons pas traité la première page de cet environnement multicols).

```
    \unvbox\partial@page
```

Puis nous retournons les marques, première et du bas, et le matériel réuni, dans la liste verticale principale.

```
    \return@nonemptymark{first}\kept@firstmark
    \return@nonemptymark{bot}\kept@botmark
    \page@sofar
```

Nous avons besoin d'ajouter une pénalité à cet endroit, ce qui nous permettra de couper en ce point puisque l'appel de la routine de sortie peut avoir supprimé l'unique point de rupture possible, « collant » ainsi tout saut suivant à la boîte équilibrée. Dans le cas où il y a des réglages bizarres de `\multicolsep`, etc., cela pourrait produire des résultats amusants.

```
    \penalty\z@
}
```

Comme nous le savons déjà, la réinsertion des notes de bas de page se fait par la macro `\endmulticols`.

La macro réalise maintenant le véritable équilibrage.

```
\def\balance@columns{%
```

Nous commençons par régler les marques conservées en les mettant à jour avec les marques de cette boîte. Cela doit être réalisé *avant* d'ajouter la pénalité de -10000 en haut de la boîte, sinon seule une boîte vide sera prise en compte.

```
    \get@keptmarks\mult@box
```

Nous continuons ensuite par la remise à zéro des essais pour enlever tout le matériel effaçable à la fin de `\mult@box`. Ceci est plutôt expérimental. Nous ajoutons aussi un point de rupture forcé au tout début, et nous pouvons donc diviser la boîte de hauteur zéro plus tard, ajoutant ainsi une glue connue `\splittopskip` au début.

```
    \setbox\mult@box\vbox{%
         \penalty-\@M
         \unvbox\mult@box
         \remove@discardable@items
         }%
```

Puis suivent des assignements de valeurs pour rendre `\vsplit` correct. Nous utilisons la partie naturelle de `\topskip` comme la partie naturelle de `\splittopskip` et permettons un peu de retrait et de dépassement par l'ajout d'élargissement et de rétrécissement.

```
    \@tempdima\topskip
    \splittopskip\@tempdima
        \@plus\multicolundershoot
        \@minus\multicolovershoot
    \splitmaxdepth\maxdepth
```

L'étape suivante est un peu délicate : quand TeX assemble le matériel dans une boîte, la première ligne n'est pas précédée par la glue d'interligne, *i.e.* il n'y a pas de paremètre tel que `\boxtopskip` dans TeX. Cela signifie que la ligne de base de la première ligne de notre boîte est située en un point imprévisible qui dépend de la hauteur du caractère le plus large de cette ligne. Mais, bien sûr, nous voulons que toutes les colonnes soient correctement alignées sur la ligne de base de leur première ligne. Pour cette raison, nous avons ouvert `\mult@box` avec un `\penalty` de `-10000`. Cela nous permettra alors de détacher un petit morceau de `\mult@box` (en fait, rien, puisque le premier point de rupture possible est le premier item dans la boîte). En conséquence, `\splittopskip` est inséré au début de `\mult@box` ce qui est exactement ce que nous voulons faire.

```
  \setbox\@tempboxa\vsplit\mult@box to\z@
```

Ensuite nous essayons de trouver un point de départ convenable pour le calcul de hauteur de colonne. Il doit être inférieur à la hauteur finale désirée mais suffisant pour atteindre cette valeur finale en quelques itérations seulement. La formule qui est maintenant implémentée essaiera de commencer avec la valeur la plus proche d'un multiple de `\baselineskip`. Le codage est légèrement subtil dans TEX et il y a peut-être un meilleur moyen...

```
  \@tempdima\ht\mult@box
  \advance\@tempdima\dp\mult@box
  \divide\@tempdima\col@number
```

Le code ci-dessus règle `\@tempdima` à la longueur d'une colonne si nous divisons simplement la boîte entière en parties égales. Pour donner le plus petit multiple suivant de `\baselineskip`, nous convertissons cette dimension en un nombre (le nombre de points d'échelle) et puis, nous multiplions ce résultat par `\baselineskip` assignant cette valeur à `\dimen@`. Cela rend `\dimen@` $\leq$ à `\@tempdimena`.

```
  \count@\@tempdima
  \divide\count@\baselineskip
  \dimen@\count@\baselineskip
```

L'étape suivante est de corriger notre résultat en prenant en compte la différence entre `\topskip` et `\baselineskip`. Nous commençons par ajouter `\topskip` ; si cela donne un résultat trop grand, alors il nous faut soustraire une `\baselineskip`.

```
  \advance\dimen@\topskip
  \ifdim \dimen@ >\@tempdima
    \advance\dimen@-\baselineskip
  \fi
```

À la demande de l'utilisateur, nous commençons avec la valeur la plus haute (ou la plus faible, mais, d'habitude, cela augmente le nombre d'essais).

```
  \advance\dimen@\c@unbalance\baselineskip
```

Nous visualisons les statistiques si cela a été demandé.

```
  \mult@info\@ne
    {Balance columns\on@line:
      \ifnum\c@unbalance=\z@\else
     (off balance=\number\c@unbalance)\fi
    \@gobbletwo}%
```

Mais nous ne permettons pas de valeurs absurdes pour commencer.

```
  \ifnum\dimen@<\topskip
    \mult@info\@ne
      {Start value
         \the\dimen@  \space ->
         \the\topskip \space (corrected)}%
    \dimen@\topskip
  \fi
```

Maintenant, nous essayons de trouver la hauteur finale des colonnes. Nous commençons par régler `\vbadness` à l'infini (*i.e.* 10000) pour supprimer les boîtes insuffisamment remplies pendant que nous sommes en train de trouver une solution acceptable. Nous n'avons pas besoin de faire cela dans un groupe puisqu'à la fin de la routine de sortie, tout est restauré. Le réglage des colonnes finales produira toujours des boîtes peu remplies, avec une « laideur » de 10000 et donc il n'y a pas lieu d'en avertir l'utilisateur.

```
  \vbadness\@M
```

Nous permettons aussi les boîtes débordantes pendant que nous essayons de séparer les colonnes.

```
  \vfuzz \col@number\baselineskip
```

La variable `\last@try` prendra la dimension utilisée dans le précédent test de séparation. Nous l'initialisons avec une valeur négative.

```
  \last@try-\p@
  \loop
```

Afin de ne pas encombrer la précieuse mémoire principale de TEX avec des choses dont nous n'avons pas besoin, nous vidons tous les registres de boîte utilisés globalement. C'est nécessaire si nous revenons à ce point après un essai infructueux. Nous utilisons `\process@cols` dans ce but, commençant avec `\mult@grightbox`. Notez les accolades supplémentaires autour de l'appel de macro : elles sont nécessaires puisque le mécanisme `\loop...\repeat` de PLAIN TEX ne peut être encapsulé au même niveau de groupement.

```
   {\process@cols\mult@grightbox
          {\global\setbox\count@
                  \box\voidb@x}}%
```

Le contenu de la boîte `\mult@box` est maintenant copié globalement dans la boîte `\mult@grightbox` (ce sera la colonne la plus à droite, comme nous devrions le voir plus loin).

```
   \global\setbox\mult@grightbox
          \copy\mult@box
```

Nous commençons par supposer que l'essai sera fructueux. Si nous finissons avec une trop mauvaise solution, nous réglons `too@bad` à `true`.

```
⟨*badness⟩
    \global\too@badfalse
⟨/badness⟩
```

En utilisant `\vsplit`, nous extrayons les autres colonnes du registre de boîte `\mult@grightbox`. Cela laisse le registre de boîte `\mult@box` intact afin de pouvoir recommencer si cet essai a été infructueux.

```
    {\process@cols\mult@firstbox{%
          \global\setbox\count@
          \vsplit\mult@grightbox to\dimen@
```

Après toutes les séparations, nous vérifions la laideur des colonnes résultantes : normalement, c'est la quantité de blanc supplémentaire dans la colonne.

```
⟨*badness⟩
          \ifnum\c@tracingmulticols>\@ne
            \@tempcnta\count@
            \advance\@tempcnta-\mult@grightbox
            \divide\@tempcnta \tw@
            \message{^^JColumn
               \number\@tempcnta\space
                badness: \the\badness\space}%
          \fi
```

Si la laideur est plus grande que celle autorisée par la laideur des colonnes, nous rejetons cette solution en fixant `too@bad` à `true`.

```
          \ifnum\badness>\c@columnbadness
            \ifnum\c@tracingmulticols>\@ne
              \message{too bad
                   (>\the\c@columnbadness)}%
            \fi
            \global\too@badtrue
          \fi
⟨/badness⟩
                         }}%
```

Il y a ici une subtilité : alors que toute les autres boîtes construites ont une profondeur déterminée par `\splitmaxdepth`, la dernière boîte aura une profondeur naturelle ne tenant pas compte du réglage original et de la valeur de `\splitmaxdepth` ou `\boxmaxdepth`. Cela signifie que nous pouvons terminer avec une profondeur très grande dans la boîte `\mult@grightbox`, ce qui pourrait donner un résultat de test incorrect. Donc, nous modifions la valeur en vidant la boîte dans elle-même.

```
    \boxmaxdepth\maxdepth
    \global\setbox\mult@grightbox
       \vbox{\unvbox\mult@grightbox}%
```

Nous sauvegardons aussi une copie de la valeur « naturelle » `\mult@firstbox` pour un usage ultérieur.

```
    \setbox\mult@nat@firstbox
       \vbox{\unvcopy\mult@firstbox}%
```

Après que `\process@cols` a fait son travail, nous sommes dans la situation suivante :

boîte `\mult@rightbox`	$\longleftarrow$	tout le matériel
boîte `\mult@gfirstbox`	$\longleftarrow$	première colonne
boîte `\mult@gfirstbox` $+ 2$	$\longleftarrow$	seconde colonne
$\vdots$		$\vdots$
boîte `\mult@grightbox`	$\longleftarrow$	dernière colonne

Nous rapportons la hauteur de la première colonne, la taille naturelle étant donnée entre crochets.

```
    \ifnum\c@tracingmulticols>\@ne
       \message{^^JFirst column
           = \the\dimen@\space
           (\the\ht\mult@nat@firstbox)}\fi
```

Si `\raggedcolumns` est forcé, les anciennes versions de ce fichier retrécissent aussi la première colonne jusqu'à sa valeur naturelle à cet endroit. Cela était réalisé de telle sorte que la première colonne ne soit pas plus petite comparée à la dernière, mais cela produisait en fait des résultats incorrects (superposition de texte) dans les cas liés. C'est pourquoi depuis la v1.5q, `\raggedcolumns` permettent à toutes les colonnes d'être légèrement courtes.

```
%    \ifshr@nking
%      \global\setbox\mult@firstbox
%              \copy\mult@nat@firstbox
%    \fi
```

Puis, nous donnons des informations sur la dernière colonne [17].

```
    \ifnum\c@tracingmulticols>\@ne
      \message{<> last column =
                \the\ht\mult@grightbox^^J}%
```

Du code de traçage que nous ne compilons pas dans la version de production à moins qu'il ne soit demandé. Il produira d'énormes listes de boîtes impliquées dans l'équilibrage dans le fichier log.

```
⟨*debug⟩
      \ifnum\c@tracingmulticols>4
         {\showoutput
          \batchmode
          \process@cols\@ne
           {\showbox\count@}}%
          \errorstopmode
      \fi
⟨/debug⟩
    \fi
```

[17] Avec TEX version 3.141, il est maintenant possible d'utiliser `\newlinechar` de LATEX dans la commande `\message`, mais les utilisateurs d'anciennes versions de TEX obtiendront désormais `^^J` plutôt qu'une nouvelle ligne à l'écran

Nous vérifions que notre essai est réussi. Le test utilisé est très simple : nous comparons simplement la première et la dernière colonne. Ainsi, les colonnes intermédiaires peuvent être plus longues que la première si `\raggedcolumns` est utilisé. Si la colonne la plus à droite est plus longue que la premère, alors nous recommençons avec une plus grande valeur de `\dimen@`.

```
    \ifdim\ht\mult@grightbox >\dimen@
```

Si la hauteur de la dernière boîte est trop grande, nous signalons cet essai comme infructueux.

```
⟨*badness⟩
      \too@badtrue
      \ifnum\c@tracingmulticols>\@ne
         \typeout{Rejected: last
                  column too large!}%
      \fi
    \else
```

Pour nous assurer qu'il n'y a pas de rupture forcée dans la dernière colonne, nous essayons d'extraire une boîte de dimension `\maxdimen` de `\mult@grightbox` (ou mieux, d'une de ses copies). Cela devrait produire une boîte vide après l'extraction, à moins qu'il n'y eut une rupture forcée quelque part dans la colonne, dans la cas où le matériel après la rupture serait resté dans la boîte.

```
⟨*colbreak⟩
      \setbox\@tempboxa
            \copy\mult@grightbox
      \setbox\z@\vsplit\@tempboxa to\maxdimen
      \ifvoid\@tempboxa
⟨/colbreak⟩
```

Donc, si `\@tempboxa` est vide nous avons une solution valide. Dans ce cas, nous prêtons un regard attentif à la dernière colonne pour décider si cette colonne doit être aussi longue que les autres ou si elle peut se permettre dêtre plus courte. Pour cela, nous devons tout d'abord remettre en boîte la colonne dans une boîte de hauteur adéquate. Si le traçage est activé, nous montrons alors la laideur de cette boîte.

```
        \global\setbox\mult@grightbox
           \vbox to\dimen@
               {\unvbox\mult@grightbox}%
        \ifnum\c@tracingmulticols>\@ne
          \message{Final badness:
                     \the\badness}%
        \fi
```

Nous comparons ensuite cette laideur avec la médiocrité autorisée pour la colonne finale. Si elle n'excède pas cette valeur, nous utilisons la boîte, sinon nous remettons en boîte une fois de plus et ajoutons de la glue au début.

```
        \ifnum\badness>\c@finalcolumnbadness
          \global\setbox\mult@grightbox
           \vbox to\dimen@
               {\unvbox\mult@grightbox\vfill}%
           \ifnum\c@tracingmulticols>\@ne
             \message{ setting natural
               (> \the\c@finalcolumnbadness)}%
           \fi
        \fi
```

Si `\@tempboxa` ci-dessus n'est pas vide, notre essai est infructueux, nous rapportons donc ce fait et réessayons.

```
⟨*colbreak⟩
      \else
        \too@badtrue
        \ifnum\c@tracingmulticols>\@ne
          \typeout{Rejected: unprocessed
            forced break(s) in last column!}%
        \fi
      \fi
    \fi
⟨/colbreak⟩
```

Si la hauteur naturelle de la première boîte est plus petite que la taille donnée par l'essai en cours, mais plus grande que celle donnée par l'essai précédent, il est probable que nous avons oublié une solution potentiellement meilleure (cela peut arriver si, pour quelque raison, la taille obtenue après notre premier essai était trop grande). Dans ce cas, nous écartons cet essai et redémarrons un nouvel essai avec la taille naturelle.

```
    \ifdim\ht\mult@nat@firstbox<\dimen@
      \ifdim\ht\mult@nat@firstbox>\last@try
        \too@badtrue
        \ifnum\c@tracingmulticols>\@ne
           \typeout{Retry: using natural
                    height of first column!}%
        \fi
        \dimen@\ht\mult@nat@firstbox
        \last@try\dimen@
        \advance\dimen@-\p@
      \fi
    \fi
```

Finalement, le commutateur `too@bad` est testé. S'il a été fixé à vrai, soit plus tôt, soit à cause d'une colonne de droite trop grande, nous essayons de nouveau avec une valeur de `\dimen@` légèrement plus grande.

```
    \iftoo@bad
⟨/badness⟩
      \advance\dimen@\p@
    \repeat
```

À ce moment, `\dimen@` prend la hauteur qui a été déterminée par la boucle d'équilibrage. Si cette hauteur de colonne devient plus grande que l'espace disponible (qui est `\@colroom`) nous faisons entrer de force les colonnes dans l'espace, en supposant qu'elles auront assez de contractibilité pour le faire[18].

```
    \ifdim\dimen@>\@colroom
       \dimen@\@colroom
    \fi
```

Puis, nous déplaçons le contenu des registres de boîte impairs vers les registres pairs, en l'étrécissant s'il le faut. Nous devons utiliser `\vbox` et non `\vtop` (comme c'était fait dans les premières versions) puisque, sinon, les boîtes résultantes n'auront pas de hauteur (*TEXbook* page 81). Cela devrait dire que le `\topskip` supplémentaire est ajouté quand les boîtes sont rendues au *page_builder* via `\page@sofar`.

```
    \process@cols\mult@rightbox
        {\@tempcnta\count@
         \advance\@tempcnta\@ne
         \setbox\count@\vbox to\dimen@
            {%
             \vskip \z@
               \@plus-\multicolundershoot
               \@minus-\multicolovershoot
             \unvbox\@tempcnta
             \ifshr@nking\vfill\fi}}%
}
```

4.5 Les allocations de boîtes

Les précédentes versions de ces macros utilisaient les premiers registres de boîte 0, 2, 4, ... pour les boîtes globales et 1, 3, 5, ... pour les boîtes locales correspondantes (Vous pourriez encore trouver des traces de ce setup dans la documentation. *soupir*). Cela pose un problème à partir du moment où nous avons plus de 5 colonnes puisque les boîtes officiellement allouées seront réécrites par l'algorithme. La nouvelle version utilise maintenant des registres de boîte privés.

```
\newbox\mult@rightbox
\newbox\mult@grightbox
\newbox\mult@gfirstbox
\newbox\mult@firstbox
\newbox\@tempa\newbox\@tempa
\newbox\@tempa\newbox\@tempa
\newbox\@tempa\newbox\@tempa
\newbox\@tempa\newbox\@tempa
\newbox\@tempa\newbox\@tempa
\newbox\@tempa\newbox\@tempa
\newbox\@tempa\newbox\@tempa
\newbox\@tempa\newbox\@tempa
\newbox\@tempa
\let\@tempa\relax
```

5 Nouvelles macros et bidouilles pour la version 1.2

Si nous n'utilisons pas TEX 3.0, `\emergencystretch` n'est pas défini, donc, dans ce cas, nous ajoutons simplement un registre ⟨*dimen*⟩ non utilisé.

```
\@ifundefined{emergencystretch}
     {\newdimen\emergencystretch}{}
```

Mes tests montrent que la formule suivante fonctionne assez bien. Néanmoins, la macro `\setemergencystretch` prend aussi `\hsize` en second argument pour permettre à l'utilisateur d'essayer différentes formules.

```
\def\setemergencystretch#1#2{%
   \emergencystretch 4pt
   \multiply\emergencystretch#1}
```

Même si cela doit être utilisé comme point d'entrée, nous utilisons `@` dans le nom, puisque c'est plus pour des experts.

```
\def\set@floatcmds{%
  \let\@dblfloat\@dbflt
  \def\end@dblfloat{\par
   \vskip\z@
   \egroup
   \color@endbox
   \@largefloatcheck
   \outer@nobreak
```

C'est facile (de différer les flottants jusqu'après la page courante) mais toute autre solution obligerait à s'enfoncer dans la routine de sortie de LATEX, et je n'aime pas y travailler tant que je ne sais pas quelle partie de la routine de sortie devra être réimplémentée pour LATEX3.

```
   \ifnum\@floatpenalty<\z@
```

Nous devons ajouter le flottant à `\@deferlist` parce que nous supposons qu'en dehors de l'environnement multicols, nous sommes en mode simple colonne. Ce n'est pas totalement correct, j'ai déjà utilisé l'environnement multicols dans une déclaration

[18] Ce sera peut être mauvais, puisque la contractibilité prise en compte pour la quantité de matériel ne pourrait être disponible que pour certaines colonnes. Mais il est mieux de le tester que de le faire directement

`\twocolumn` de LaTeX, mais cela ira pour beaucoup d'applications.

```
      \@cons\@deferlist\@currbox
    \fi
    \ifnum\@floatpenalty=-\@Mii
      \@Esphack
    \fi}}
```

5.1 Maintenance des registres de marque

Cette section contient les routines qui fixent les marques pour qu'elles soient prises correctement en compte. Elles ont été introduites avec la version 1.4.

La première chose que nous faisons est de réserver trois noms de macro, pour prendre le texte de remplacement, pour les primitives de TeX, `\firstmark`, `\botmark` et `\topmark`. Nous initialisons les deux premières pour qu'elles soient vides, et `\kept@topmark` qui contient deux paires d'accolades vides. C'est nécessaire puisque `\kept@topmark` est supposé contenir la dernière marque provenant d'une page précédente, et, dans LaTeX, toute marque « réelle » doit contenir deux parties, représentant les informations des marques droites et gauches.

```
\def\kept@topmark{{}{}}
\let\kept@firstmark\@empty
\let\kept@botmark\@empty
```

Quelquefois, nous voulons retourner la valeur d'une marque « conservée » dans un nœud `\mark` de la liste verticale principale. C'est réalisé par la fonction `\return@nonemptymark`. Comme le nom le suggère, elle agit seulement si le texte de remplacement de la marque conservée n'est pas vide. C'est fait pour éviter d'ajouter une marque vide, quand aucune marque n'est effectivement présente. Si, néanmoins, nous voulons ajouter une telle marque, elle doit être considérée plus tard comme une `\firstmark` valide.

```
\def\return@nonemptymark#1#2{%
  \ifx#2\@empty
  \else
```

Dans un but de débuggage, nous jetons un coup d'œil à la valeur de la marque conservée que nous voulons retourner. Ce code le fera.

```
<*marktrace>
      \mult@info\tw@
        {Returned #1 mark:\MessageBreak
         \meaning#2}%
%       \nobreak
%     \fi
</marktrace>
```

Puisque le contenu de la marque peut être un code LaTeX arbitraire, nous nous assurons mieux de son incapacité à entraîner une expansion quelconque (des expansions ont été déjà réalisées pendant l'exécution de `\markright` ou `\markboth`). Nous utilisons donc le mécanisme habituel d'un registre de token pour interdire l'expansion[19].

```
      \toks@\expandafter{#2}%
      \mark{\the\toks@}%
```

Nous ne voulons aucun point de rupture entre une telle marque retournée et le matériel suivant (qui, habituellement, est juste la boîte d'où venait la marque).

```
      \nobreak
    \fi}
```

Si nous avons du matériel dans un registre de boîte, nous voulons, peut-être, mettre la première et la dernière marques hors de cette boîte. Cela peut être réalisé par `\get@keptmarks` qui prend un argument : le numéro du registre de boîte ou son alias, défini par `\newbox`.

```
\def\get@keptmarks#1{%
```

Dans un but de débuggage, nous regardons les dimensions courantes de la boîte car dans les anciennes versions du code, j'ai fait quelque erreurs dans ce domaine.

```
<*debug>
       \typeout{Mark box #1 before:
                ht \the\ht#1, dp \the\dp#1}%
</debug>
```

Maintenant, nous ouvrons un nouveau groupe et, localement, nous y copions la boîte. En conséquence, toute opération, *i. e.* `\vsplit`, auront seulement un effet local. Sans cette astuce, le contenu de la boîte perdrait le niveau auquel le dernier assignement du registre de boîte a été réalisé.

```
  \begingroup
    \vbadness\@M
    \setbox#1\copy#1%
```

Maintenant, nous étendons la boîte jusqu'à la dimension maximale possible. Cela doit séparer le contenu entier de la boîte donc, tout est réellement séparé. Par conséquent, `\splitfirstmark` et `\splitbotmark` contiendront respectivement la première et la dernière marque.

```
    \setbox#1\vsplit#1to\maxdimen
```

[19] À cause de la définition courante de `\markright`, etc., ce ne serait pas utile de définir la commande `\protect` pour interdire l'expansion, puisque tout `\protect` a déjà disparu à cause des expansions précédentes

Donc, nous pouvons maintenant fixer les marques conservées, ce qui est une opération globale, et, ensuite, fermer le groupe. Cela restaurera le contenu original de la boîte.

```
  \set@keptmarks
\endgroup
```

Pour débugger, nous regardons encore la dimension de la boîte qui ne doit pas avoir changé.

```
⟨*debug⟩
   \typeout{Mark box #1 \space after:
            ht \the\ht#1, dp \the\dp#1}%
⟨/debug⟩
}
```

La macro **`\set@keptmarks`** est responsable du réglage de **`\kept@firstmark`** et **`\kept@botmark`**, en vérifiant la valeur courante de **`\splitfirstmark`** et **`\splitbotmark`**.

```
\def\set@keptmarks{%
```

Si **`\kept@firstmark`** est vide, nous supposons qu'elle n'est pas configurée. Strictement parlant, cela n'est pas correct puisque nous avons perdu la capacité d'avoir des marques explicitement vides, mais pour les applications normales de LaTeX, cela est suffisant. Si elle n'est pas vide, nous ne changeons pas la valeur – ensuite, dans les routines de sortie, elle sera restaurée comme **`\@empty`**.

```
  \ifx\kept@firstmark\@empty
```

Nous mettons maintenant le contenu de **`\splitfirstmark`** dans **`\kept@firstmark`**. Dans le cas où il n'y avait pas du tout de marque, **`\kept@firstmark`** ne sera pas modifiée par cette opération.

```
    \expandafter\gdef\expandafter
       \kept@firstmark
       \expandafter{\splitfirstmark}%
```

Au cours du débuggage, nous montrons les assignements mais seulement si quelque chose se produit réellement.

```
⟨*marktrace⟩
    \ifx\kept@firstmark\@empty\else
      \mult@info\tw@
        {Set kept first mark:\MessageBreak
         \meaning\kept@firstmark%
         \@gobbletwo}%
    \fi
⟨/marktrace⟩
  \fi
```

Nous essayons toujours de fixer la marque du bas à **`\splitbotmark`** mais, bien sûr, seulement quand il y a **`\splitbotmark`**. De nouveau, nous supposons que si **`\splitbotmark`** est vide, cela signifie que la partie séparée de la boîte ne contient plus de marque.

```
  \expandafter\def\expandafter\@tempa
     \expandafter{\splitbotmark}%
  \ifx\@tempa\@empty\else
     \global\let\kept@botmark\@tempa
⟨*marktrace⟩
     \mult@info\tw@
       {Set kept bot mark:\MessageBreak
        \meaning\kept@botmark%
        \@gobbletwo}%
⟨/marktrace⟩
  \fi}%
```

La fonction **`\prep@keptmarks`** est utilisée pour initialiser les marques conservées à partir du contenu de **`\partial@page`**, *i. e.* la boîte qui prend tout à partir du haut de la page courante avant de démarrer l'environnement multicols. Cependant, une telle boîte est valable seulement si nous ne produisons pas de multicols emboîté.

```
\def\prep@keptmarks{%
  \if@boxedmulticols \else
    \get@keptmarks\partial@page
  \fi}
```

```
\def\remove@discardable@items{%
⟨*debug⟩
  \edef\@tempa{s=\the\lastskip,
               p=\the\lastpenalty,
               k=\the\lastkern}%
  \typeout\@tempa
⟨/debug⟩
       \unskip\unpenalty\unkern
⟨*debug⟩
  \edef\@tempa{s=\the\lastskip,
               p=\the\lastpenalty,
               k=\the\lastkern}%
  \typeout\@tempa
⟨/debug⟩
       \unskip\unpenalty\unkern
⟨*debug⟩
  \edef\@tempa{s=\the\lastskip,
               p=\the\lastpenalty,
               k=\the\lastkern}%
  \typeout\@tempa
⟨/debug⟩
       \unskip\unpenalty\unkern
⟨*debug⟩
  \edef\@tempa{s=\the\lastskip,
               p=\the\lastpenalty,
               k=\the\lastkern}%
  \typeout\@tempa
⟨/debug⟩
       \unskip\unpenalty\unkern
}
```

```
⟨*badness⟩
\newif\iftoo@bad
```

```
\newcount\c@columnbadness
\c@columnbadness=10000
\newcount\c@finalcolumnbadness
\c@finalcolumnbadness=9999

\newdimen\last@try

\newdimen\multicolovershoot
\multicolovershoot=2pt
\newdimen\multicolundershoot
\multicolundershoot=2pt
\newbox\mult@nat@firstbox
⟨/badness⟩
```

Une aide pour produire les messages d'information.

```
\def\mult@info#1#2{%
  \ifnum\c@tracingmulticols>#1%
   \GenericWarning
        {(multicol)\@spaces\@spaces}%
        {Package multicol: #2}%
  \fi
}
```

6 Réglage de `\columnwidth`

Si nous stockons la largeur de la colonne courante dans `\columnwidth`, nous devons redéfinir la macro interne `\@footnotetext` pour qu'elle utilise la largeur des notes de bas de page, plutôt que la définition originale.

Depuis la v1.5r, cela est maintenant réalisé de telle façon que la définition originale soit toujours utilisée, sauf que, localement, `\columnwidth` est fixé à `\textwidth`.

Cela résoud deux problèmes : premièrement, les redéfinitions de `\@footnotetext` réalisées par une classe seront toujours valides ; deuxièmement, si multicols est utilisé dans un environnement minipage, la définition spéciale de `\@footnotetext` dans cet environnement sera prise et non celle du texte principal en cours de traitement (ce dernier devrait entraîner la perte de toutes les notes de bas de page dans ce cas).

Voir la définition de la commande `\multicols` pour le code exact.

7 Extensions supplémentaires

Cette section contient le code pour les extensions ajoutées à ce package. Toutes peuvent être actives, certaines peuvent être dormantes et en attente de leur activation dans de futures versions.

7.1 Non-équilibrage des colonnes

C'est assez facile à implémenter. Nous avons juste à inactiver la routine de sortie d'équilibrage et à la remplacer par celle qui expédie les autres pages.

Le code pour cet environnement a été proposé par Matthias Clasen.

```
⟨*nobalance⟩
 \@namedef{multicols*}{%
```

Si nous sommes dans le texte principal en cours de traitement, *i. e.* dans une boîte de cette sorte, cette approche ne fonctionnera pas, puisque nous n'avons pas de dimension verticale, il est donc préférable d'avertir l'utilisateur plutôt que de réaliser l'équilibrage.

```
   \ifinner
     \PackageWarning{multicol}%
       {multicols* inside a box does
        not make sense.\MessageBreak
        Going to balance anyway}%
   \else
     \let\balance@columns@out
          \multi@column@out
   \fi
   \begin{multicols}
}
```

Quand l'environnement est fini, nous fermons simplement l'environnement `multicols` intérieur, sauf qu'il vaut mieux, également, être dans un glue étirable verticalement, pour que la dernière colonne, qui contient toujours le texte, ne soit pas étirée verticalement de manière exagérée.

```
 \@namedef{endmulticols*}{\vfill
   \end{multicols}}
⟨/nobalance⟩
```

7.2 Rupture manuelle des colonnes

Le problème, avec les ruptures manuelles de page dans multicols est que, pendant la collecte du matériel pour toutes les colonnes, une pénalité de « forçage de page » (*i. e.* -10000 ou plus) pourrait arrêter la passe de collecte, ce qui n'est pas tout à fait ce qui est désiré. D'un autre côté, l'utilisation d'une pénalité telle que -9999 signifierait qu'il y aurait des cas pour lesquels les opérations `\vsplit` dans multicols ignorerait une telle pénalité et choisirait encore un point de rupture différent.

Pour cela, l'implémentation actuelle utilise une approche totalement différente. En un mot, elle étend le traitement de la routine de sortie de LaTeX en introduisant un drapeau de pénalité supplémentaire (*i. e.* un pénalité qui est forcée mais supérieure à -10000, pour que la routine de sortie puisse considérer cette valeur et donc, savoir pourquoi elle a été appelée).

À l'intérieur de la routine de sortie, nous testons cette valeur et si elle apparaît, nous faisons deux choses : sauvegarde du texte en cours de traitement à partir de ce point dans une boîte spéciale pour une utilisation ultérieure et réduction de `\vsize` de la hauteur du matériel vu. De cette façon, la pénalité est maintenant cachée dans cette boîte et nous pouvons redémarrer le processus de collecte pour les colonnes restantes (cela est réalisé par `\speci@ls` ci-dessus).

Dans les routines de sortie qui appliquent `\vsplit` soit pour l'équilibrage soit pour une page entière, nous combinons simplement la boîte 255 et la boîte sauvegardée, ce qui donne donc une boîte unique de séparation, qui contient maintenant les ruptures forcées à la bonne place.

`\columnbreak` est construit d'après `\pagebreak`, sauf que nous produisons une pénalité de -10005.

```
⟨*colbreak⟩
\mathchardef\@Mv=10005
\def\columnbreak{%
```

Nous devons nous assurer qu'elle est utilisée seulement à l'intérieur d'un environnement multicols car si nous avons une pénalité pouvant être considérée par la routine de sortie inchangée de LaTeX, des choses étranges pourraient apparaître.

```
\ifnum\col@number<\tw@
 \PackageError{multicol}%
  {\noexpand\columnbreak outside multicols}%
  {This command can only be used within
   a multicols or multicols* environment.}%
\else
 \ifvmode
   \penalty -\@Mv\relax
 \else
   \@bsphack
   \vadjust{\penalty -\@Mv\relax}%
   \@esphack
 \fi
\fi}
```

Nécessite une boîte pour collecter le texte en cours de traitement à partir de la rupture de colonne.

```
\newbox\colbreak@box
⟨/colbreak⟩
⟨/package⟩
```

Index

Numbers written in italic refer to the page where the corresponding entry is described, the ones underlined to the code line of the definition, the rest to the code lines where the entry is used.

Change History

v1.0c

\enough@room : Penalité 0 ajoutée pour vider la liste de contribution. 9

v1.0d

General : Toutes les lignes sont raccourcies à 72 ou moins 1

v1.0e

General : Redéfinition de l'env. description pour utiliser \descriptionmargin=5pt dans la documentation 1

\prepare@multicols : \textwidth changé en \linewidth. 12

Fixation de \columnwidth supprimée. 12

v1.0f

General : \z@ fixé à 0 pt pour la redéfinition de description 1

v1.1a

General : \multicolssep changé en \multicolsep. 1

\flushcolumns : \flushedcolumns renommé en \flushcolumns. 19

v1.2a

\balance@columns : Suppression du groupe autour de la boucle principale. 20

\prepare@multicols : \pretolerance -1 car ça ne réussit presque jamais. 12

\set@floatcmds ajouté. 12

\setemergencystretch ajouté. 12

\vbadness 10001 maintenant. 12

\set@floatcmds : Macro ajoutée. 23

\setemergencystretch : Macro ajoutée. 23

\speci@ls : Boîtes flottantes libérées. 18

v1.3a

\balance@columns : Changement de \vtop en \vbox. 23

v1.3b

\endmulticols : Réalise \penalty avec \addpenalty 13

\enough@room : Réalise \penalty avec \addpenalty 10

\multicols : Minimum de deux colonnes 8

v1.3c

\balance@columns : \global\advance remis à partir de l'ancien code 22

Limitation de la hauteur de colonne à \@colroom 23

\endmulticols : Vérification de la fermeture de l'env. 14