

Journal d'évolution d'Onversity 2008

(version : Luxor)

Pour discuter de la nouvelle version ou poser des questions, recomposez l'adresse e-mail suivante :

Luxor-onversity.net (remplacer '-' par '@') ;

Samedi 29 Novembre

Re-indexing a database with variable size field and record : (This text is extract from the whitepaper : Databases engines – A global solution for Varchar)

Why re-indexing? The reason of re-indexing came mostly from two cases. First you want to delete physically records that are already marked as deleted. Secondly, the database has a problem and it usually came from the index. Therefore re-indexing will regenerating all index and provide a solution to the problem.

The first thing you do when re-indexing a database with fixed record size is to check the master file with the equation:

Nbrs of records * Size of record = Master file size (measure are in bytes)
(The master file is where all the data are store except index.)

In our case, that checking is a dead end because we can't know for sure the number of records from the master file. The only way we have if to list / count / read every chars, every field and every record. The time needed is not that long (complexity is O(n)). In fact this is a good thing because you have to check that each record has exactly the numbers of field requested.

It is a matter of fact, with static record and field size database, we usually admitted that when a record is written in the master file, the size of this record is correct. The equation is a check of this reality. That assumption is fully acceptable. The probability that a write order is always fulfill is very high. Rather the write order (usually a record) is written into the storage device rather it's not, but having a partial write is merely to happen.

With variable record size we can't assume that because there isn't a unique size for each record. We have to act as if a partial write has happened. How it's working?

You load a block of n bytes of the master file (where $n > \text{max size of a record}$). You count each end of string (char(0)) which represent the number of field. But how do we know for sure that the field number x (for example) we have analyzing belong to the same record. Couldn't be that the field x is in fact the field one of the next record ? This can happen with a partially written record.

The solution to that little problem is to use a marker, a beacon for each record. I will make it short. One good way to do this is to have a field associated to the record position. The lower the

number is, the earliest the record has been recorded. With that solution you know that between record number 17001 and 17002 there must be x field. This solution applied for any record 'y and y+1'.

To be continued...

Maquiné Jean-François

Ps : The algorithm of "A global solution to Varchar" is based on a fully running database engine, yet not revealed to the public. If you're a corporate you may want to know more about the algorithm. Contact me, so we can meet.

Email: luxor at onversity.net (replace " at " by "@")

From the same author:

The D-Loop Pattern : A technique for optimizing loops containing conditional branches (text in English)

<http://www.onversity.com/load/d-loop.pdf>

Analytical solution to the mean distance covered by hard drives heads (classical solution use probabilities – text in French)

http://www.onversity.net/cgi-bin/progarti/art_aff.cgi?Eudo=cYkWlraie&P=a0801

Jeudi 27 Novembre

I'm back : bon ben voilà j'étais en vacances je n'ai pas touché de codes pendant presque 3 semaines, après une bêta très hard, puisqu'il s'agissait d'une alpha et donc les bogues ont plu ! La deuxième phase a démarré et consiste finalement à concevoir les forums, permettre la gestion et l'affichage des news et des édits. Il faudra que je revienne sur des éléments de la phase 1 aussi. Au final, l'alpha test devrait démarrer en février au plus tard.

Forums, forums, ... : Les nouveaux forums seront très différents des anciens. Présentation radicalement différente mais qui se démarque de tout ce qui existe. Accès aux commandes plus simple et mieux pensé et surtout plus de fonctions pour l'administrer. Modération, suppression, déplacement, ...

Mardi 9 Octobre

Enfin ... : J'ai enfin pu me dire aujourd'hui mais que dois-je faire ☺. Bref tous les scripts sont là et il n'y plus que du contrôle à faire, pour l'essentiel. Je dois avouer que la messagerie interne était bien plus complexe à mettre en œuvre que prévu. Par exemple pour écrire un message, il faut tester l'existence de l'auteur, du destinataire, que leurs boîtes à message ne sont pas pleines, qu'il autorise bien les messages privés, il faut écrire le message dans la base, mettre à jour le compteur de message pour l'auteur et celui de nouveau message pour le destinataire.

La bêta 1 sera la version 0.2 :000. Il y aura 3 autres bêtas et non 2. Je suis sur les genoux voilà la raison. Après chaque bêta je prendrais du repos. Mais la bêta 1 était certainement la plus grosse bêta.

Les contraintes du Xhtml 1 strict et du Html 5 : Dans le cas du Xhtml 1 strict, on ne peut plus ouvrir une fenêtre à partir d'une fenêtre, sauf à passer par le javascript. L'attribut target= _blank n'est plus. De même les inputs des formulaires doivent être intégrés dans des box types <p> ou <div>. Pour le HTML 5, on oublie les cellpadding et cellspacing. Il faut regarder du côté du CSS avec border-spacing et border-collapse pour résoudre les problèmes que pose l'absence de ces deux attributs.

Mercredi 24 septembre

Silence, silence : Il n'y a pas grand-chose à ajouter en ce moment, je finalise la première bêta. Je suis donc confronté à une tonne de petits détails pas très motivants, mais bon ça doit être fait. Le stress est à son comble et je m'interroge pour savoir comment le serveur va accepter toutes les nouvelles technologies, si la programmation 64 bits est bonne et comment les utilisateurs vont percevoir la nouvelle version.

Le travail de cette semaine consiste essentiellement à faire le ménage dans le code et à faire des skins. Les maquettes sont faites, mais il faut les décomposer dans des fichiers graphiques exploitables par le code CSS. Il faut aussi faire modifier un peu le code CSS (les couleurs par exemple).

Programmation 64 bits : je ne dispose pas actuellement de station de développement 64 bits, je fais donc avec des données théoriques et les schémas d'implantation du 64 bits sur Windows et Linux. Windows utilise LLP64 et Linux LP64. Les deux conservent les int en 32 bits. Seuls changent les longs. Ils ont donc tous été retirés et ne sont plus utilisés pour assurer une compatibilité 32/64 bits Windows/Linux. Toutefois à terme l'utilisation directement de registre 64 bits peut permettre des optimisations. Effectivement un microprocesseur optimisé 64 bits préfère travailler avec des registres 64 bits que 32 bits. Je pense en particulier aux compteurs de boucle et aux pointeurs de tableaux sans compter les variables de la structure des bases de données d'Hydrogen.

Le fichier de log des erreurs : Hydrogen est doté depuis la version 1 d'un fichier de log des erreurs. La version 2 l'a considérablement étendu et peut à présent servir à déboguer les scripts qui utilisent les fonctions d'Hydrogen. Dès que l'un d'eux utilise de manière curieuse les fonctions, un message est écrit dans le fichier log. En remontant le message je peux déterminer assez facilement la raison du problème. À plusieurs reprises, durant ces neuf dernières années, les bases d'Hydrogen ont été confrontées à des problèmes. Beaucoup peuvent être éliminés si on encadre bien l'utilisation des fonctions. Par exemple une clef vide lors d'une fonction de recherche doit être détectée avant de lancer la fonction de recherche afin de ne pas l'exécuter.

Mercredi 10 septembre

Un grand jour pour la science : A 10h27 le LHC a produit ses premières collisions. Bien qu'on exagère un peu ses défis, le LHC pourrait soit confirmer de manière importante les théories physiques, soit les mettre profondément à mal. En particulier, il faut trouver le boson de Higgs, une particule qui donnerait leur masse aux autres particules. Stephen Hawking a parié 100\$ qu'on ne le trouvera pas, ça serait plus amusant pour lui si on ne le trouvait pas. Je dois dire que je suis assez de cet avis, mais on serait vraiment très mal pour pas mal d'années. Il faudrait peut-être même envisager de doper les scientifiques pour que leur cerveau fonctionne à un rendement plus élevé.

Un peu plus tard, le LHC en panne pour 6 mois au moins !

Samedi 6 septembre

Un silence apparent : Juste pour vous dire que je travaille dur, que ça avance malheureusement lentement. Je suis sur des éléments dont je ne peux discuter ouvertement. La concurrence existe bel et bien et il n'est pas question que je parle de solution technique alors que j'ai encore plusieurs mois avant la sortie de la nouvelle version. La bêta test est toujours maintenue pour d'ici 3 à 5 semaines. J'espère terminer la gestion des utilisateurs la semaine qui vient. Après j'attaque un autre pan d'Onversity à réécrire et tester tout cela en mode 32 bits et 64 bits sur le serveur et on pourra attaquer. 3 semaines c'est faisable. La semaine prochaine je lancerais la bêta test sur le site. La tête que vont faire les bêta testeurs :D

Mercredi 27 août

Beta-testeurs recherchés : Vous rêviez de devenir beta-testeur sur Onversity ? Onversity réalise votre rêve. Je sais que plus d'un souhaiterait s'inscrire à cette bêta-teste juste pour voir à quoi va elle va ressembler, mais ce n'est vraiment pas le bon plan, pour vous et pour Onversity. Voici les profils :

- Personne sérieuse et méticuleuse
- Système d'exploitation Linux, Apple, Windows
- Animé d'une véritable volonté d'aider à produire une nouvelle version d'Onversity d'un niveau professionnel avec des technologies avancées.
- Aucune connaissance particulière ne sera requise.

Il s'agit en réalité d'une alpha-test. L'objectif est de tester les technologies de base en condition réelle, d'évaluer l'interface homme ainsi que la présentation. Elle ne porte que sur environ 1/3 des scripts.

Les bêta-testeurs devront

- Se conformer aux étapes de tests

- À chaque étape exécuter une check-list qui leur sera confiée. Rendre un rapport de bogue en rapport avec la check_list
- Chercher à reproduire les bogues rencontrés

C'est un vrai job, bien qu'il ne soit pas payé, les personnes participant aux tests entiers auront des compensations (non monétaire).

A partir de la parution de ce test, il faudra encore 3 à 4 semaines pour terminer et disposer sur le serveur de la version de tests. Semaine 1 : ménage et déboguage, semaine 2 : Skin, semaine 3 : compilation sur le serveur en 32 bits et 64 bits, ménage du code déboguage , skin. Semaine 4 : on sait jamais ;).

La première alpha-test devrait commencer d'ici 3 à 4 semaines. Des informations complémentaires seront données lors du contact. Seront choisies de préférence des personnes connues sur Onversity.

Pour vous faire connaître : luxor-onversity.net (remplacez le – par le @).

Vendredi 22 août

Toujours sur la page dictionnaires : Encore 2 semaines de boulot sur cette page ... et beaucoup aura été fait.

Adieu Usemap : Les Usemap sont des graphiques découpés en zone et chaque zone correspond à un lien internet. Les Usemap sont utilisés à travers le HTML. Grâce à eux on peut concevoir un menu présenté dans une image. Cela permet de faire des menus plutôt sympas et surtout de disposer des polices de caractères de votre choix. Le problème est que les usemap ce n'est pas compatible avec le navigateur pour malvoyant et ce n'est plus dans l'air du temps. Les moteurs de recherche ne savent pas quoi en faire.

Les menus vont donc se transformer en des listes `<a>..`. C'est même déjà fait.

La fin d'un mystère : Durant les dernières années j'ai eu à plusieurs reprises des utilisateurs qui m'ont averti de leur impossibilité de valider certains formulaires. Il semble que j'ai trouvé la raison et elle est inattendue. Le renvoi d'informations n'est pas toujours fait de la même manière. L'analyse des données des formulaires prend maintenant en compte la variabilité dans le renvoi des informations. J'espère que ce sera suffisant.

De la compatibilité d'IE7 : Il est courant de dire qu'Internet Explorer ne respecte pas les normes W3C. Il est vrai qu'il est moins performant que Safari 3, Firefox et Opera 9.5. Toutefois utilisant les dernières techniques CSS et Xhtml, j'ai pu aussi constater qu'il est moins psychorigide que ses concurrents dans certains domaines. Avec IE7 certains de mes écarts à la norme passent, pas avec les autres.

De l'utilité des normes il n'est pas question ici, mais à vouloir imposer des normes à la virgule près ça frise la névrose. Tout développeur d'interface homme-machine sait qu'il doit prendre en compte la variabilité des réactions face à l'interface qu'il développe pour ne pas rendre fou une partie de ses utilisateurs. Au nom de quoi les théoriciens des normes Internet s'autorisent-ils à imposer une application aussi stricte. Les esprits rigides ne sont pas les meilleurs garants des bonnes évolutions.

Quoiqu'il en soit Onversity n'aura aucun problème à utiliser IE7 en Xhtml et CSS.

Jeudi 14 août

Xhtml et CSS : Après avoir acquis les connaissances pour faire du code CSS et Xhtml conforme aux dernières normes les plus strictes et ce pour les quatre principaux navigateurs, je me suis lancé dans les niveaux supérieurs. On a d'une part l'établissement de règles simples et strictes sur l'utilisation de différentes balises. Cela répond à des questions comme à quel moment utilise-t-on des balises P et non div. Des balises Hn au lieu de P. Doit-on déclarer un Css text-size dans un div ? Quelle doit être la forme d'un clear :both. Quand utiliser les box de type float et position. Par exemple, après un travail de recherche et de test il s'avère que les tables ne sont à utiliser que lorsque l'utilisation des cellules est complexe ou lors de l'utilisation de bordure interne, thanks to border-collapse setting. Ces règles viennent d'être établies après quatre jours de tests des différentes possibilités (mais je préparais cela depuis 2 ans en fait). D'autre part il me faut affiner ma connaissance de l'utilisation d'héritage dans l'utilisation des classes et la différenciation claire d'utilisation entre id, style et class. À cela s'ajoute la suppression de toutes les tailles fixes. Pour ceux qui s'interrogent sur l'existence de style dans une page elle est simple. Un auteur peut décider d'utiliser du CSS pour son texte, il le fera par les styles et non les id ou les classes, ceux-ci étant dans un fichier à part.

Utilisation de DOM et javascript. Dans la mesure du possible leur utilisation sera limitée. Garder Onversity simple au niveau développement est la principale règle. Toutefois on se permettra un peu de DHTML mais essentiellement avec du CSS la possibilité à laquelle je pense se confirme.

Dimanche 10 août

Chevauchement : Les dictionnaires avancent et avec eux quasiment toutes les technologies de base d'Onversity. J'ai même trouvé des 'petits trucs' amusants. Mais ça sera pour plus tard. Pour finaliser les dictionnaires il me faut intégrer la gestion des tags. ça pourrait être 'simple', mais il y a concurrence avec la base des syntagmes. Effectivement, ADN permet de trouver une liste de mots pertinents pour chaque texte. Cette liste est enregistrée de plusieurs manières pour être utilisée par différents algorithmes. Le problème est que les syntagmes peuvent tantôt être assimilés à cette liste de mots et en être une extension, tantôt en être totalement indépendants.

La particularité des syntagmes est qu'ils sont considérés comme obligatoirement représentatifs du texte. C'est leurs définitions. Cela permet, blablabla ... j'en dis trop... blablabla...

Je dois simplifier les technologies pas seulement pour l'utilisateur, mais aussi pour moi. Il me faut trouver un moyen de faire coexister tag et liste d'ADN simplement.

Tout ça pour dire que j'en suis toujours à travailler sur des technologies de bases d'Onversity. De base signifie qui peuvent être utilisées par n'importe quel script si nécessaire.

Mercredi 30 juillet

Premier résultat d'ADN : Appliqué aux bases de données des dictionnaires, ADN fournit un résultat rapide et plus pertinent que son ancêtre nommé Oxyde_SE.EN à un moment. ADN trouve 13 500 syntagmes composés de 1 à 6 mots contre 17 500 syntagmes composés de 1 à 5 mots.

De l'utilité de D-loop : Un des Spin-off de D-loop, une technique d'optimisation des boucles que j'ai mise en point en 2004, est une fonction Strlen haute vitesse. Elle a juste une contrainte d'utilisation car il peut lui arriver de dépasser pour 1 octet la taille d'allocation. Il y a une exception à cette règle si la chaîne de caractères se termine par un double zéro (pour le langage C). Or c'est le cas des enregistrements d'Hydrogène 2.

Lorsqu'on lit un enregistrement, il faut ensuite affecter un pointeur à chaque champ car on ne connaît pas la taille des champs. Il faut la déterminer à chaque lecture. La fonction qui fait ça est

```
i= 1 ;
pos = 0;
*chp_pos = chp;
while(i < Bs->nbchp)
{
    while(*(chp + pos) != '\0')
    {
        pos++;
    }
    pos++;
    *(chp_pos + i) = (chp + pos);
    i++;
}
```

Qui devient

```
i= 1 ;
pos = 0;
*chp_pos = chp;
while(i < Bs->nbchp)
{
    pos += (new_strlen((chp + pos)) + 1);
}
```

```

        *(chp_pos + i) = (chp + pos);
        i++;
    }

```

Et New_strlen est :

```

size_t new_strlen(const char * src)
{
    unsigned int i = 0;

    while ((unsigned char) src[i] != '\0')
    {
        if ((unsigned char) src[i + 1] == '\0')
        {
            i++;
            break;
        }
        i += 2;
        while (((unsigned char) src[i] & (unsigned char) src[i + 1]) != 0)
            i += 2;
    }

    return i;
}

```

Mardi 29 juillet

Parlons pointeur : La correction de l'algorithme principal d'Hydrogène 2 est en bonne voie. Comme je l'ai précédemment expliqué, cela a amené à supprimer la gestion de mémoire cache pour les fonctions d'écriture. C'est un type de révision que je gardais pour la version 2.5 d'Hydrogène. Je suis maintenant dans l'obligation de réfléchir sur certains éléments ayant trait à la version 2.5. Parmi ceux-ci il y a les pointeurs. Avec la version 2 est apparu un nouveau type de pointeur pour Hydrogène, les 'Char ***', c'est-à-dire des tableaux de tableaux de pointeurs de chaînes de caractères. Cela correspond au pointeur de position de chaque champ d'un enregistrement. Il y a N enregistrements ayant M champ ayant chacun une chaîne de L caractères. Toutefois ce qui nous intéresse à un moment donné c'est le 'Char **', c'est-à-dire le tableau de chaîne de caractères d'un enregistrement donné, ou dit autrement le tableau des pointeurs vers chaque champ d'un enregistrement. Or c'est justement ça qui nous intéresse dans les fonctions sans cache et uniquement cela. Effectivement il n'est plus question d'avoir un tableau de l'ensemble des enregistrements. Seul l'enregistrement en cours nous intéresse.

J'ai donc pu supprimer toute référence à des 'Char ***' dans les fonctions en écriture. Est-ce une bonne chose puisque l'écriture et l'utilisation des fonctions avec et sans cache deviennent très différentes et non interchangeable. J'avoue que je ne le sais pas encore. Peut-être est-ce nécessaire de rationaliser cela, je n'en sais rien, mais si je rationalise, cela va compliquer inutilement les fonctions sans cache qui vont devenir majoritaires. Il se pourrait même que je doive écrire de suite les fonctions de lecture sans cache. J'aurais ainsi quasiment écrit la

version 2.5 et perdu un temps précieux.

Pour finir voyons la différence entre un 'char **chp[1]' et 'Char **chp', Fondamentalement le [1] ne sert à rien pour manipuler le tableau de pointeur. Par contre il permet d'affecter un char ** à un autre char ** en passant par une fonction.

```
Fo1 (char **chp)
{
    // x = enregistrement numéro x
    Cache[x] = chp ;
}
```

```
Fo2 (char ***chp)
{
    // x = enregistrement numéro x
    Cache[x] = *chp ;
}
```

```
Main()
{
    Char **chp1 ;
    Char **chp2[1]

    Fo1(chp1) ;
    Fo2(chp2) ;
}
```

L'appel à Fo2 est correct pas l'appel de Fo1. Il y a plusieurs raisons. La première est que vous ne pouvez pas passer en paramètre un pointeur non déclaré. Ensuite le programme va sauvegarder l'adresse de chp1 lors du passage de paramètre annulant l'affectation faite à chp dans Fo1.

Pour conclure. La gestion des pointeurs entre des fonctions ayant le même rôle avec gestion des caches et sans, diffère considérablement et bien plus que je ne m'y attendais. De fait, sachant que la version 2.5 devait amener cette révision de création de fonction avec et sans gestion des caches, et que cela amènera à revoir tous les scripts, il serait peut-être judicieux de créer les fonctions de lecture sans cache dès la version 2.0. Cela m'amènera à réviser tout ce qui avait déjà été fait. Soit 1 mois de travail au moins. La génération précédente d'Hydrogène n'avait pas ce problème, c'est-à-dire une différence si marquée entre fonctionnement des fonctions sans cache et avec. C'est pour cette raison que j'ai pensé pouvoir intégrer cette modification dans une version 2.5. En réalité, la majorité des fonctions appelant les bases de données n'ont pas besoin d'une gestion des caches.

Dimanche 27 juillet

On a frôlé la catastrophe : Jeudi 24 juillet j'écrivais que Hydrogène 2 faisait fort. Vendredi 25

juillet, Hydrogène 2 s'écroulait suite à son incapacité à trier correctement les index. Pas d'index, pas de moteur de base de données.

Au début je pensais qu'il s'agissait juste d'un problème suite à une modification qui visait à normaliser l'utilisation des 'unsigned' et 'signed' dans Hydrogène. J'ai ensuite essayé quelques modifications simples, cela solutionnait partiellement le problème mais pas définitivement. J'ai ôté les doigts du clavier et je suis retourné à la théorie.

Pour comprendre le problème d'Hydrogène 2, il faut comprendre ce qui s'est passé lors du passage de la génération 1 d'Hydrogène à la génération 2. Dans la première génération, les index sont composés d'un unique entier pour chaque enregistrement. Cet entier contient deux informations. La première est le numéro d'enregistrement de l'enregistrement du fichier maître, la seconde est la position physique dans le fichier maître de l'enregistrement. Une donnée, 2 informations. Il faut bien saisir que cela est permis car la taille des enregistrements est fixe et on a la formule :

$$\text{Position physique} = \text{Numéro d'enregistrement} * \text{taille de l'enregistrement}$$

Avec des enregistrements de taille variable, cette petite, mais au combien très utile équation, n'a plus de raison d'être. Petit rappel pour mieux comprendre la suite. La position physique est l'information qui nous permet de lire l'enregistrement sur le disque dur. Le numéro d'enregistrement est ce qui permet de placer et de retrouver l'enregistrement dans les tableaux de stockage provisoire.

L'une des idées qui vient à l'esprit est d'ajouter l'information manquante au index. Donc chaque index ne serait plus constitué d'un entier, mais de deux. L'un pour la position physique de l'enregistrement l'autre pour le numéro d'enregistrement. Problème ! Cela double quasiment le temps de traitement des données tant en lecture qu'en écriture. En particulier toucher au temps de lecture d'Hydrogène va à l'encontre de sa conception.

En fait le problème avec Hydrogène est l'écriture. Pour la lecture la forme première était bonne. Dans ce contexte une autre solution apparaît. Celle de faire une table de stockage de fichier maître par index. C'est assez volumineux mais dans un seul cas. L'écriture séquentielle d'information. Je ne vais pas entrer dans le détail de cette idée. Sachez juste qu'elle est basée sur la réflexion suivante. Pourquoi a-t-on besoin d'index ? Pour éviter de devoir trier directement le fichier maître. Celui-ci étant très volumineux cela prendrait beaucoup plus de temps à le trier. Mais on peut le faire avec les tableaux de stockage provisoire dans la mesure où on ne déplace que des pointeurs sur des données.

Poussons l'idée précédente plus loin. Pourquoi a-t-on besoin de cache ? Pour augmenter la vitesse de traitement du moteur de recherche. Peut-on s'en passer ? Dans beaucoup de situations où, en particulier lorsque les enregistrements lus ne le sont qu'une seule fois, ce qui est le cas dans la recherche d'existence d'un élément ou la lecture séquentielle d'une base. Au final l'idée est de simplement supprimer les fonctions de cache pour l'écriture. Pas de cache pas besoin du numéro d'enregistrement, pas de surcharge de la mémoire.

C'est cette solution qui a été retenue. Elle oblige par contre à faire quelque chose (au moins partiellement) que j'aurais souhaité faire dans la version 2.5 d'Hydrogène, revoir la gestion des caches et construire des fonctions avec et sans cache. La suppression des capacités de cache en écriture va m'obliger à dupliquer et modifier des fonctions cruciales et pour le moins très

difficiles à mettre en œuvre. Mais là je n'ai pas le choix !

En résumé : Le passage d'Hydrogène 1.X à 2.0 a fait perdre une information qui était vitale à la gestion des caches. En conséquence lors de l'écriture, Hydrogène 2.0 ne traitait pas correctement les nouveaux enregistrements. La solution consiste à supprimer la gestion des caches en écriture. Cela va impliquer d'intégrer une séparation entre fonction en écriture et fonction en lecture. En terme de performances la suppression des caches en écriture est négligeable car on insère un élément après l'autre dans la majorité des cas et ces cas là n'ont quasiment aucun recours au cache. Dans le pire des cas on pourra se reposer sur les caches de l'OS.

Jeudi 24 juillet

Hydrogène 2.0 fait fort : Pour tester les algorithmes du nouveau moteur de recherche il me faut des bases de données. Je suis en train de faire la conversion des anciens dictionnaires. Le

dictionnaire informatique avec 450 définitions passe de 1877 ko à 371 ko. La gestion variable de la taille des enregistrements porte ses fruits. Mais je ne m'attendais pas à une telle différence dans les dictionnaires.

Skin en folie : J'ai été pris à deux reprises d'une 'skin-mania'. J'ai fait skin sur skin. On arrive aujourd'hui à un total de 16 skins ! Il faut que je me calme. J'ai hâte de voir vos têtes quand vous verrez la nouvelle version (N'oubliez pas de prendre une photo :D). Parmi, les nouvelles skins, je citerais : Apple (impressionnant celui-là), Firefox, Green & Zen, Porsche GT3-RS (très sport), mosaïques (indéfinissable mais certains vont adorer). Grand changement pour la skin de base. J'avais fait une skin classique qui reprenait les couleurs vert, bleu, gris d'Onversity, mais avec des tons moins austères. J'abandonne cette idée et les couleurs de base seront celles de la skin 'Cerenkov effect'. Vous verrez, c'est très joli. Par contre pour les nostalgiques il y a une skin 'Originale'.

Mercredi 23 juillet

SE.EN 3.0 : L'équation de principe qui permet de s'affranchir d'un tri par date ou par occurrence est écrite. Il faudra de nombreux réglages, mais bon elle est là. A priori SE.EN 3.0 sera une surcouche à SE.EN 2.0. La transition se fera en douceur et plus d'options seront disponibles.

Centralisation des textes : L'une des particularités de SE.EN est de pouvoir sauvegarder les recherches. La fonction est plus une version alpha qu'une version aboutie. Elle sera reprise dans SE.EN 2.0. Pour faire cette sauvegarde et pouvoir la manipuler, les données des textes d'Onversity sont centralisées. Dans la version actuelle cette centralisation, bien qu'existante, est imparfaite. Grâce à une centralisation pensée depuis le départ, la fonction de sauvegarde devrait être bien supérieure à la version actuelle et surtout plus utile. En pratique, cette centralisation correspond à une base de données qui regroupe des informations comme le type de texte, ses dates, son domaine d'études, sa difficulté, ... mais pas le texte lui-même. La base contenant le texte contient le texte et un lien vers le fichier centralisé. Je résume un peu car c'est quand même un peu plus complexe que cela pour des raisons d'optimisation.

C'est parti mon kiki : Bon ben maintenant que j'ai pris mes vacances et réfléchi sur la forme que devait avoir SE.EN 2.0, j'ai plus qu'à le faire. Ma première victime sera les dictionnaires. Je vais faire en même temps, les dictionnaires, le moteur de recherche et la gestion YOUM vous savez la fenêtre qui s'ouvre quand vous cliquez sur un mot en bleu. Sauf que là plus de fenêtre, les fonctions YOUM seront déportées sur les pages des dicos et d'autres trucs que vous verrez quand vous y serez. Simplicité et efficacité sont les directives qui dominent cette partie de développement.

Mardi 22 juillet

SE.EN 2.0 en route : D'un point de vue utilisateur, l'existence d'un moteur de recherche sur un site comme Onversity n'est pas une priorité. En commençant Luxor je le pensais aussi. Hélas, au fil du développement cette idée a commencé à s'effriter et quand j'ai commencé les dictionnaires 1 mois avant de partir en vacances, l'idée a fait plouf en tombant à l'eau.

L'outil fondamental des moteurs de recherche d'Onversity est ADN, le séquenceur de textes. J'ai donc terminé ADN, au passage comme YOUM 3 partage de nombreux algorithmes avec ADN j'étais bon pour une révision en profondeur de YOUM 3 mais au passage je me suis préoccupé de ses performances. ADN et YOUM ne sont pas de ces algorithmes qu'on peut quitter puis revenir dessus 6 mois plus tard sans y passer plusieurs jours avant de faire un bon travail dessus. L'optimisation de YOUM 3 aurait pu attendre mais comme j'étais dans le code, ç'aurait été une perte de temps de ne pas le faire.

Je dois à présent réévaluer les deux moteurs de recherche que se partage la version actuelle et les fusionner, réfléchir sur les fonctions du moteur de recherche. J'espère que je n'en aurais pas pour trop longtemps mais ce travail préliminaire est essentiel si je veux limiter les risques de devoir reprendre beaucoup de code car je me serais trompé dans mon analyse. Les premières conséquences de mes réflexions sont :

- ADN devra fournir non seulement une liste de mots pertinent d'un texte mais les occurrences de ces mots. Cela ne modifie pas l'algorithme il s'agit juste d'un ajout de fonction au moment où l'on ajoute un mot à la table des mots produits.
- Les résultats d'un moteur de recherche sont basiquement soit triés par les occurrences des mots recherchés dans les textes du site, soit par date. Toutefois cela est très limité car un vieux texte comportant de nombreuses références à un des mots cherchés peut être plus intéressant qu'un texte récent qui y fait une seule fois référence. J'ai commencé à travailler sur un algorithme qui résout ce problème. Je n'en dirais pas plus, mais je vais concevoir SE.EN 2.0 pour recevoir cet algorithme quand j'aurais le temps de le développer et qu'il aura mûri dans ma tête.
- La structure des fichiers est en cours d'écriture. À cela s'ajoutera un fichier qui résout une faille du système d'informations par occurrence. À nouveau je n'en dirais pas plus mais il faut distinguer ce qui est une information globale d'une information locale. Ce fichier sera utile avec l'algorithme avancé de SE.EN 2.0 qui pourrait bien prendre le nom de SE.EN 3.0.

Lundi 21 juillet

ADN est fini : ADN produit aujourd'hui une liste de mots très épurée, bien supérieure à son prédécesseur et plus rapide. ADN est très beau et sexy comme algorithme.

YOUM 3 est optimisé : Une des conséquences de la révision en profondeur d'ADN a été d'optimiser YOUM 3. Le séquenceur d'ADN est la base du code et les principales fonctions sont totalement identiques. C'est un grand progrès de YOUM 3 de normaliser à ce point le code. Les performances de YOUM 3 sont très proches de YOUM 2 sur processeur Intel Conroe et devraient être similaires sur processeur Penryn grâce à l'optimisation de la division dont dispose ce dernier. YOUM 2 disposant d'une optimisation software pour la division, ne l'utilise pas et donc ne bénéficie pas de l'optimisation hardware. Le développement de YOUM 3 a atteint TOUS les objectifs que j'imaginai à son début. C'est un beau succès.

Samedi 20 juin

ADN avance : Le séquenceur d'Onversity qui vient de recevoir le nom de séquenceur ADN (depuis le temps que je l'utilise !). ADN était déjà implanté dans YOUM3, mais je n'avais fait que contrôler son fonctionnement de base. Là je me suis attaqué à son optimisation et à son débogage par un contrôle sur les 1080 actualités. Pour ADN optimisation ça veut avant tout dire qualité de la liste des syntagmes produits, puis vitesse de production de la liste. Toutefois il s'agit d'un équilibre parfois subtil. Ça prend énormément de temps. Dans tous les cas le séquenceur de Luxor est terriblement plus performant que celui de la version actuelle d'Onversity.

Un peu d'ADN pour YOUM 3 : Par l'optimisation d'ADN, YOUM 3 s'en est retrouvé tout ragaillardi. Les problèmes de sous-performance que je rencontrais avec les compilateurs autres que ceux d'Intel se sont envolés et les performances de YOUM 2 et 3 sont proches. Il faudra attendre la fin d'optimisation d'ADN puis de YOUM 3 pour savoir ce qu'il est réellement.

Samedi 7 juin

Qu'est-ce que la normalisation du code : Certains l'auront remarqué, j'attache beaucoup d'importance à la normalisation du code pour la version Luxor. C'est une des raisons qui m'oblige à reprendre de manière systématique chaque bout de code déjà écrit pour le réécrire et cela prend beaucoup de temps. Que recouvre réellement ce terme ? Il s'agit de faire en sorte que la majorité du code de programmation utilise les mêmes techniques algorithmiques que cela soit pour l'accès aux bases de données, la déclaration des variables, la manière de nommer les variables, la gestion des dates, etc... Pour rendre efficace la normalisation, on lui associe deux techniques. La première consiste dans le fait qu'un code utilisé par plusieurs autres bouts de code soit centralisé dans une librairie et généralisé pour être utilisé par le maximum d'autres bouts de code. Plus que la réutilisation, c'est la centralisation du code qui est important ici. Cela permet de passer plus de temps pour écrire correctement les fonctions centralisées au lieu de passer du temps sur beaucoup de bouts de code qui se ressemblent même si ceux-ci sont plus optimisés en termes de performance. La seconde est de rendre dynamique la gestion des tailles des variables et les définir sur des variables liées aux bases de données. J'ai encore trouvé un bogue dans la version actuelle d'Onversity à cause d'une variable dont la taille est insuffisante. Si elle avait été indexée sur la taille de son champ et normalisée pour le travail qu'elle devait faire, ce bogue n'existerait pas.

Pour dire les choses autrement, la normalisation c'est l'établissement de règles simples et suffisamment générales à l'écriture de code et qu'il faut respecter dans toutes les parties même si cela occasionne des ralentissements. La normalisation c'est l'industrialisation du code et celle-ci est nécessaire dès que vous atteignez un certain volume de code. C'est depuis longtemps le cas avec Onversity, mais il fallait tout reprendre pour pouvoir pratiquer la normalisation du code. D'où l'existence de Luxor.

Jeudi 5 juin

LE dictionnaire ... : J'avais oublié à quel point les dictionnaires étaient le coeur d'Onversity. Je suis en train péniblement de terminer la fonction surprise pour les dictionnaires. J'ai corrigé bogue après bogue durant 2 semaines. J'ai du mettre en oeuvre de nouveaux types d'outils. Tout ce qui concerne les dictionnaires va être regroupé sur une même page. Cela n'a pas été sans poser des problèmes de lisibilité, mais même si tout n'est pas terminé cela devrait plaire et ne ressemble à rien de connu. Ce sera surtout très pratique. J'allais en dire plus, mais non :).

Pourquoi LE dictionnaire ? D'une part tous les dictionnaires ont été regroupés dans la même base et d'autre part, j'ai regroupé sur une même page d'administration la saisie des dictionnaires et des données de bases de données associées au dictionnaire.

ADN : Vous connaissez YOUM et SE.EN, respectivement les technologies générant à partir de textes des liens vers les dictionnaires en temps réel et le moteur de recherche d'Onversity. En fait ces deux technologies se basent sur un séquenceur de syntagme que j'ai toujours aussi appelé SE.EN. Pour plus de clarté, ce qui est un thème central de Luxor, j'ai décidé de nommer ce séquenceur ADN.

Pour pouvoir générer les dictionnaires, il me faut ADN. Pour avoir ADN, il me faut extraire tous les éléments générant des liens dans YOUM, car YOUM est construit sur ADN. C'est un peu de travail mais sans plus. J'ai l'habitude de cette tâche même si YOUM 3 diffère grandement de YOUM 2. Il me faudra quand même déboguer un peu plus YOUM 3.

YOUM : Une fois ADN fait et les bases de données des dictionnaires version Luxor opérationnelles, j'intégrerais YOUM 3 à Luxor. Pour l'instant YOUM 3 n'a été conçu et testé que pour fonctionner avec l'ancienne version d'Onversity. Un peu de travail mais ça devrait aller.

SEEN : Une fois ADN et YOUM opérationnels avec des bases de données représentatives, je pourrais m'attaquer à SE.EN 2.0, le nouveau moteur de recherche.

Actuellement, il existe en fait 2 moteurs de recherche sur Onversity. Il va falloir que je les réanalyse pour déterminer lequel choisir, mais j'ai déjà ma petite idée. Tout deux sont des moteurs statistiques basés sur les résultats d'ADN et comptent de manière très différente. De plus ils sont semi-dynamiques. Cela signifie que j'ai fait une grosse ânerie lors de la mise en place des moteurs de recherche sur Onversity. A l'origine ils devraient être entièrement dynamique, c'est-à-dire qu'ils devraient régénérer toutes leurs statistiques et compiler tous les textes à chaque recherche. Impossible. D'un point de vue algorithmique ce n'est pas tenable. J'ai persisté dans cette idée et j'ai fait un système avec des statistiques en partie situées dans une base et en partie calculées en temps réel. En fait j'étais tellement fier des performances du séquenceur de syntagmes que je n'ai pas voulu abandonner l'idée d'une analyse temps réel.

Dans le nouveau moteur de recherche, l'analyse se basera sur une base de données statistique qui sera mise à jour soit manuellement par l'administrateur, soit à chaque ajout de document. Il y aura aussi une petite différence dans la manière de compter et ce qu'on compte.

Mise à plat : Tout ceci, dictionnaires, ADN, YOUM, SE.EN, m'a obligé à prendre du temps

pour mettre à plat toutes ces technologies et à bien réfléchir à leur intégration dans Luxor. Une conséquence est que je reporte la réalisation des QCM et des Forums tant que ce travail d'intégration de technologies ne sera pas fait.

Samedi 31 mai

Homogénéisation des entrées des scripts : En script CGI il existe deux méthodes pour passer des paramètres d'un script à un autre. La méthode GET est celle qui est utilisée naturellement lorsqu'on redirige un script vers un autre automatiquement et avec des paramètres, de même pour les scripts accessibles par des liens sur lesquels il faut cliquer. La méthode POST est celle naturellement utilisée pour le transfert des paramètres d'un formulaire d'un script à un autre. D'un point de vue technique, la méthode POST est adaptée aux paramètres de grande taille comme un texte de forums, et aux mots de passe puisqu'on ne voit pas dans l'url les valeurs des variables. Si mes souvenirs sont bons des tests que j'ai effectués il y a 8 ans pour comprendre les différences entre GET et POST. La méthode GET est limitée dans la taille de son buffer de stockage de variable, mais est plus rapide (par exemple avec la méthode GET on ne lit qu'une variable d'environnement au lieu de deux pour la méthode POST).

Ceci étant dit, dans la version actuelle les deux méthodes cohabitent (ce qui est normal), mais utilisent toutes deux des algorithmes très différents pour accéder aux variables. En particulier la méthode POST utilise une fonction d'Onversity pour dispatcher le buffer d'entrée vers les différentes variables. Celles-ci sont stockés dans un tableau de chaîne de caractères contrôlé par un tableau qui définit pour chaque script, le nombre de paramètres et leur taille maximum respective. Cela permet un contrôle des allocations mémoires, des buffer overrun et le surplus ou le manque de paramètres.

Pour la méthode GET, j'utilise pour affecter les données du buffer d'entrée un simple Sscanf et des noms de variable spécifique. C'est un reliquat d'un bout de code que j'ai trouvé en 1999 quand j'ai commencé à apprendre la programmation des CGI en langage C. Toutefois la fonction Sscanf ne supporte pas les espaces dans un variable. Cela implique donc de faire une analyse des codes spéciaux après le Sscanf ce qui n'est pas cohérent avec la méthode POST. Je vais donc gérer la méthode GET de la même manière que la méthode POST. Cela va homogénéiser la gestion de la fonction Main() et rationaliser les fonctions Onversity de décodage des paramètres d'entrées (Lors d'un passage d'un script à un autre les caractères spéciaux doivent être codés par un code hexadécimal) et d'affectation.

Il faudra que je reprenne tous les scripts qui passent des paramètres ne provenant pas d'un formulaire. Cela prendra encore un peu de temps, mais le jeu en vaut la chandelle.

L'homogénéisation ouvre la porte à la création d'une fonction qui serait une sorte de fonction main() d'Onversity et tous les scripts feraient appel à cette fonction qui générerait les paramètres passés ou non, les contrôles des paramètres et le lancement de la fonction Corps(), Affiche(), Ces dernières fonctions ayant été homogénéisées depuis le début. Cela signifie que le schéma de traitement d'un formulaire est le même pour tous les script, le schéma d'affichage d'une information aussi,... Mais la création d'une fonction remplaçant la fonction Main est unique pour tous les scripts, et ne sera pas pour la version Luxor. Je pourrais parfaitement l'intégrer petit à petit.

Fermeture des forums : Les forums de la version en ligne ont été fermés. J'ai autre chose à faire que de courir après des rigolos de service et pester contre le manque de fonctionnalités, de modernité des forums actuels.

Vendredi 23 mai

Liste multi-colonne des définitions : En attendant le CSS3 et sa gestion multi-colonne d'une liste d'informations, pour la simuler il faut connaître le nombre d'éléments de la liste. S'agissant d'une base de données et à moins d'avoir un champ réservé pour cela seul, la situation peut devenir compliquée. Dans l'ancien système de gestion des dictionnaires, il y avait une base de données par dictionnaire. Pour connaître le nombre d'enregistrements, il suffisait de diviser par 4 la taille d'un fichier d'index. Avec le nouveau modèle qui regroupe tous les dictionnaires cela ne fonctionne plus. Heureusement Hydrogen utilise la dichotomie tant décrite pour la gestion des bases de données. Il suffit de faire une recherche sur le premier élément d'un dictionnaire dans la base, une recherche sur le dernier, et comme les éléments d'index sont triés, il suffit de faire une simple soustraction entre la position rendue lors de la première recherche et la seconde.

Ca à l'air de rien mais cela permet de gérer un des grands succès dans la gestion des scripts d'Onversity. Ils sont rapides et consomment peu de mémoire. Sur Onversity le mécanisme est le suivant : On lit en base de données une information et on l'affiche de suite. D'une part elles sont déjà triées, il suffit de choisir le ou les bons index à l'ouverture de la base de données, d'autre part on ne stocke / copie pas les données provenant de la base. De nombreux scripts sur internet s'occupent de lire toutes les données avant affichage, puis d'afficher la liste de ces données.

Le principal avantage que je vois vu de ce côté du Rubicond est que cela permet de gérer plus facilement les messages d'erreur si l'erreur survient entre le premier et le dernier élément. J'ai résolu en partie ce problème avec la version Luxor d'Onversity en donnant la possibilité de faire appel au gestionnaire d'erreur à l'intérieur d'un script et pas seulement comme un script indépendant. Effectivement une fois que vous avez commencé à afficher une page vous ne pouvez plus faire de redirection automatique vers le gestionnaire d'erreur et il n'est pas pensable de demander à un utilisateur qui ne comprend pas ce qui arrive à l'écran de cliquer sur un lien pour voir apparaître le gestionnaire d'erreur.

Hydrogen 2.5 suite : Ce que me rappelle cette histoire c'est que toutes les fonctions d'Hydrogen sont liées à une mise en cache des données lues dans les bases. Ceci est très utile lorsqu'on travaille de manière répétitive sur des données, mais inutile pour des données lues à coup sûr une seule fois. Dans le cas des définitions, on les lit toutes une fois et une seule, mais toutes sont

mises en cache et ce cache ne sert à rien. Ce qui signifie des Mo de mémoire réservés inutilisés. Je ne peux intégrer rapidement les principales fonctions sans cache car cela nécessite une certaine réflexion pour une intégration parfaite entre les deux ensembles de fonctions. Cela devrait permettre de gagner en performance (diminution drastique du nombre d'appel aux fonctions de réservation mémoire) et dans l'utilisation de la mémoire. Évidemment cela obligera à modifier les scripts, mais cela devrait pouvoir se faire de manière compatible entre la version 2.0 et 2.5 et donc le passage pourra se faire lentement script par script sans devoir tout changer d'un coup. Les fonctions avec cache restant totalement d'actualité et restant elles intactes.

Mercredi 21 mai

32 ? Non, 48 ! : Hydrogen est conçu autour de certaines constantes. L'une d'entre elles est le nombre maximum de champ qu'une base de données peut gérer. Hydrogen 1 définit cette constante à 20. Avec Hydrogen 2 j'ai commencé à 24 pour rapidement passer à 32. Mais finalement ça sera 48. La base de données des profils n'a cessé de prendre en importance.

Hydrogen 2.5 : Hydrogen s'appelait auparavant Maqbase. Hydrogen 2 correspondant à la version 7 de Maqbase. Lors du passage de Maqbase 2 à Maqbase 3 j'ai modifié profondément la structure qui gère les bases de données pour simplifier son utilisation et la rendre plus dynamique pour consommer moins de mémoire. J'ai aussi perdu 10% en performance. Une fois que Luxor sera terminé, je prendrais le temps de voir s'il est possible de remanier cette structure pour gagner en vitesse. Il s'agit de vitesse purement processeur et mémoire n'ayant rien à voir avec les disques durs. L'organisation de cette structure influe sur la vitesse à laquelle on accède aux données en mémoire. Si j'arrive à l'améliorer alors on passera à la version 2.5 d'Hydrogen. Ce travail ne peut pas être fait pour la version 2 car cela nécessite de nombreuses recherches pour trouver la forme la plus performante. Cela peut prendre des semaines.

Vendredi 16 mai

Intéressant projet de dictionnaires : Il paraît que j'en dis trop. Donc je ne pourrais pas vous dire pourquoi je traîne sur les dictionnaires en ce moment et qu'il y en a encore pour un petit moment. Je peux quand même vous dire que c'est un vieux projet qui remonte à 4 ou 5 ans. Je l'ai toujours repoussé car il n'est pas simple techniquement à mettre en oeuvre, mais de nouvelles compétences récemment acquises facilitent la vie. De plus il posait des problèmes organisationnels auxquels j'ai pu trouver des solutions. Surprises, surprises :)

Machines arrières toutes : Il ne faut pas exagérer. Favoriser l'organisation des bases de données au détriment des performances ça va un moment mais trop c'est trop. Sur toutes les pages, il y a un rappel des principaux textes récents. Cela m'oblige donc à ouvrir la base générale qui contient tout une somme de petits détails pour chaque texte et les grosses bases d'articles, éditos, .. Bref au mieux j'ai quatre ouvertures de base et ça monte à 7 parfois. C'est un peu petit, beaucoup trop lourd. En créant dans la base générale un espace pour le texte de présentation du texte qui est le plus souvent le début du premier paragraphe, je peux, dans bons nombres des pages, n'ouvrir qu'une seule base de données. Il y aura vraisemblablement

redondance entre les grosses bases et la base générale, mais un gros gain de performances aussi.

Lundi 12 mai

Refonte des dictionnaires : J'ai commencé à faire les nouveaux dictionnaires. Il va y avoir une refonte profonde tant dans leur affichage que dans la gestion de leur base de données. La tâche sera difficile mais nécessaire comme je l'explique plus bas. En fait la difficulté n'est pas garantie car j'applique de nombreuses simplifications. Mais j'ai mis plusieurs mois pour obtenir la forme qu'ils ont actuellement. Je redoute donc un possible inconvénient. J'en saurais plus dans quelques jours

Affichage : Dans le pire des scénarios, il faut 5 clics avec la version actuelle pour accéder à une définition une fois qu'on est sur la page des dictionnaires. La présentation actuelle est sobre et soignée, mais malheureusement pas si pratique. Dans un effort d'efficacité, la nouvelle présentation affichera toujours la totalité des définitions de tous les dictionnaires. Il suffira donc d'un click pour accéder à une définition dans tous les cas. Un autre avantage serait la possible augmentation de visibilité des définitions des dictionnaires par les moteurs de recherche.

Base de données : Pour optimiser les accès aux dictionnaires surtout pour YOUM, il y a une base de données par dictionnaire. Cela évitera donc à chaque définition, par exemple informatique, de rappeler le nom du dictionnaire dans l'enregistrement. La conséquence est que le nombre de tests à faire par enregistrement lors d'une recherche est toujours égal à 1, et l'on fait l'économie d'un champ. Avec le nouveau moteur, Hydrogen 2, la taille variable des champs diminue lourdement la taille des enregistrements et donc l'idée de diminuer le nombre de champs pour diminuer la taille de l'enregistrement ne tient plus. Concernant le nombre de tests de champ lors d'une recherche, la situation est plus subtile. Cela était parfaitement vrai lorsque l'analyse d'un texte se faisait avec un seul dictionnaire à la fois. Mais depuis 3 ans, l'analyse de texte est multi-dictionnaire. Il faut donc ouvrir plusieurs bases de données pour analyser un texte même de quelques octets. Avec une base de données unique, les temps d'ouverture seraient plus courts et compenseraient en parti au moins le surplus de travail par des tests multiples de champ.

Pour finir il faut savoir que pour obtenir une grande souplesse d'utilisation des multi-dictionnaires, j'ai dû mettre au point un algorithme sophistiqué pour la librairie des bases de données. Avec une base de données unique pour les dictionnaires, on simplifie grandement l'utilisation et la maintenance à long terme.

Vendredi 9 mai

Retour sur la déficience d'IE : Je rappelle qu'IE est affecté d'une déficience dans sa gestion des caches. Il s'agit en fait de l'algorithme intelligent qui est censé déterminer quand une page doit être rechargée ou non. Cela ne prête pas à conséquence normalement sauf dans un cas, lorsque la page est un formulaire pour mettre à jour des données d'une base de données. Dans l'administration cela concerne la moitié des scripts, pour un site web cela concerne par exemple

la modification des informations de son profil ou d'un message sur les forums. C'est donc plus limité.

Microsoft est parfaitement au courant de ces problèmes depuis la version 4. Il propose des solutions exotiques, genre mettre une section <head></head> en fin de script xhtml après la balise </body>. J'ai essayé les balises meta classique censées résoudre le problème. Cela ne résout rien et je me refuse à utiliser des idioties parce que Microsoft se refuse à corriger le bogue.

Il faut bien comprendre qu'avec Microsoft la plupart des améliorations sont liées à leur valeur médiatique. Celle-ci ayant une valeur quasi nulle, on pourrait bien attendre IE 13.0 pour voir le problème corrigé.

Les solutions que je fais appliquer sont doubles. En premier dans le cas de l'administration, je n'ajouterais aucune balise particulière pour régler ce problème de cache et IE sera interdit dans l'administration. Vlan dans les dents. Pour le site on ne peut pas interdire IE. Je vais donc utiliser le truc en javascript. Celui-ci consiste à ajouter un paramètre au script. S'il est à 1 alors on écrit une page xhtml très courte qui contient un programme javascript qui force le navigateur à recharger la page.

En fait je n'aime pas cette solution je préférerais ne rien faire, mais j'ai découvert qu'avec l'ancienne version si j'applique bien l'idée d'un paramètre 0/1 je n'utilise pas de script javascript et IE fonctionne très bien. L'astuce ? Je viens de penser que je n'ai pas vérifié ce problème avec la gestion de profil du site.

J'ai encore des tests à faire, mais hier j'ai testé sur le serveur lui-même pour voir si IE réagissait différemment. Ma station de développement est sous Windows XP et IIS alors que le serveur est sous Linux et Apache. Mais rien de change.

Changement d'environnement de développement : En prévision du passage à Vista 64 bits, je vais devoir passer de mon vieux mais fidèle environnement Borland BC5 à Visual C++ 2008 Express qui est devenu gratuit. J'ai donc fait le ménage dans mes programmes pour installer Vc9 (= Vc 2008). Ça fonctionne pas mal. Toutefois le compilateur Intel 10.1 ne s'installe pas sur cette version de Vc9. J'espère que ça fonctionnera avec le compilateur Intel version 11.0

Premier pas sur le serveur d'Onversity : Pour faire les tests d'IE sur le serveur, j'ai compilé avec succès et exécuté avec succès plusieurs pans de la gestion de l'administration. Tout s'est très bien passé. Très bonne nouvelle.

Mercredi 7 mai

Un bug étonnant : Toujours aux prises avec la modernisation du code j'ai rencontré un problème avec cette ligne de code. Les connaisseurs du langage C apprécieront

```
printf(" (%s : %s)",cs("Créateur du texte",buffer),cs(*(Pchp1_pos[0] + Central_cre),buffer));
```

La fonction `cs(A,B)` transforme une chaîne A en une chaîne B et retourne B. Le problème est que le compilateur évalue d'abord les deux expressions `cs()` puis exécute `printf()`. Mais comme dans ce cas `cs()` retourne la même variable et que la dernière analysée est la première dans l'ordre d'apparition le résultat à l'affichage est :

Créateur du texte : Créateur du texte

au lieu de

Créateur du texte : jfmaquine

Évidemment la solution serait de pouvoir créer la variable à l'intérieur de la fonction appelée et non appelante. Mais on ne peut pas en langage C. Si on le fait on fout soit le bordel dans les pointeurs, soit dans la mémoire, soit dans la pile. Il faut savoir que le problème ne se pose pas de la même manière s'il s'agit d'un pointeur ou d'une variable.

Comme j'ai un peu abusé de la fonction `cs()`, je suis bon pour un contrôle d'une tonne de codes. Il commence à y en avoir :D

PS : J'ai utilisé le terme de bug parce que j'ai découvert cela en faisant une série de débogage, mais en fait il s'agit d'une limitation. Il est tout à fait envisageable d'avoir un langage qui n'a pas cette limitation, mais cela ne se ferait-il pas au détriment soit de la consommation mémoire, soit des performances ?

Jeudi 1 mai

Vers la fin de la modernisation du code : Au fil des années, j'ai eu de nombreuses occasions de voir où le code C d'Onversity pouvait être amélioré. Bon nombre de modifications avaient déjà été mises en place, mais cette semaine d'autres sont venues s'y ajouter. J'ai essentiellement trouvé des approches pour rendre le plus de variables dynamiques possibles dans la partie « `main()` » et généraliser des fonctions pouvant être utilisées par tous les scripts mais centralisées dans le module Tools 2.0.

Mais cette semaine j'en suis arrivé à un point où continuer à moderniser le code n'apporterait plus grand chose si ce n'est de retarder l'avancement de la version Luxor et donc sa mise en place. Je l'ai assez fait comme ça. Même si je trouve encore quelque chose à redire dans les semaines qui viennent, et que cela n'est pas bloquant, cela devra attendre la prochaine version.

J'ai hâte de répercuter les dernières trouvailles sur tous les scripts existants, de m'attaquer aux QCM puis aux forums. Quand cela sera fait, on pourra dire que Luxor commence à voir le bout du tunnel. 1 mois après la création des forums je devrais être en mesure de fixer une date de sortie :).

Lundi 28 avril

Des problèmes de fond refont surface : Après avoir réglé des problèmes de fond avec la

gestion des cookies et le moteur de base de données, deux nouveaux problèmes sont venus se rappeler à mon bon souvenir. Le premier est le fameux problème de mise à jour de l'affichage des scripts des formulaires. Effectivement Internet Explorer conserve un cache même si on utilise la commande

```
<meta http-equiv="cache-control" content="no-cache" />
```

ou

```
<meta http-equiv="pragma" content="no-cache" />
```

il faut les deux plus la directive

```
<meta http-equiv="expires" content="now" />
```

lire

<http://www.pacificnet.net/~johnr/meta.html>

<http://support.microsoft.com/kb/q222064/>

Lors de la création des forums en 2001, j'ai été confronté à ce problème. A l'époque comme en atteste la date des documents Microsoft, il n'y avait pas vraiment de solution. J'avais remarqué par contre que IE ne pouvait pas utiliser son cache si les paramètres changeaient. J'ai donc utilisé une commande javascript qui recharge la page et relance le script avec un paramètre load=1, le script étant appelé à la base avec la valeur load=0. C'est une solution très efficace et totalement compatible et à l'épreuve du temps. Toutefois j'ai conçu Onversity à l'origine sans prêter attention au nombre de paramètres. Avec Luxor je souhaite au contraire minimiser le nombre de paramètres des scripts. Pour l'instant la solution avec les balises meta semble fonctionner et être compatible avec les futures normes. Dans le cas contraire, je retournerais à la solution javascript et au paramètre load. A moins que quelqu'un ait une solution viable, fiable, compatible et à l'épreuve du temps.

Un problème avec la solution à base de balises meta est qu'il ne faut pas les laisser pour tous les scripts sinon l'activité du site en prend un coup de même que la bande passante du serveur puisque les navigateurs des internautes n'utilisent plus la technique de la mise en cache.

Je constate aussi malheureusement que seul IE semble touché. Safari, Opera, Firefox ne semblent pas nécessiter de balise ou un paramétrage personnalisé. Au final, la suppression de la mise en cache ne doit se faire que sur les scripts de formulaire dédiés à la modification d'informations dédiées à la modification de données.

Un autre problème est celui des codes de retour chariot dans les zones de saisie de texte de type <textarea>. Lorsqu'on saisie du texte, on utilise des retours chariot pour aérer son texte. Le problème est que lorsqu'on réaffiche le texte dans une zone textarea les codes retour chariot sont doublés. La solution que j'avais développé en 2001 était de faire une fonction de remplacement des retours chariot par des codes HTML équivalents à savoir la balise
. Lorsqu'il fallait réafficher le texte dans un textarea d'un formulaire pour modification, on faisait l'opération inverse. Ceci est nécessaire car un textarea n'analyse pas les codes HTML ils apparaissent tel quel. Je pensais avoir trouvé une solution plus simple. En fait non, et je vais reprendre la solution existante qui, même si je la trouve redondante, reste efficace. Ah oui, à l'époque j'avais

aussi constaté une différence de réaction dans les codes de retour chariot entre Windows et Linux, le remplacement par des
 avait solutionné cette différence de comportement.

Pour conclure, je suis pris dans un cycle de résolution de problèmes de fond. De vieux problèmes, et je dois réanalyser toutes les solutions que j'avais trouvées, voir comment la technologie a évolué vis-à-vis de ces problèmes et envisager d'autres solutions si possible ou nécessaire. Ah et en analysant des tonnes de codes d'autres sites, j'ai pu constater que le charset UTF-8 n'est pas si marginal que cela. Je vais donc devoir refaire une analyse de la pertinence à passer en UTF-8.

Samedi 26 avril

Spécifications du HTML 5 : ayant fait un gros travail de mise aux normes pour le code des navigateurs, je suis donc de près les futures normes CSS3 et HTML 5. Pour le CSS 3 pas de problème de compatibilité et je sais déjà les caractéristiques qui m'intéressent et qui seront exploitées sur Onversity. Pour le HTML 5 c'est encore un peu tôt. Il y a évidemment du ménage dans les balises et leurs options dont beaucoup disparaissent au profit du CSS. Mais ceci était attendu depuis longtemps donc pas de surprises. Par contre on trouve des choses gênantes. La disparition des options de tableau cellpadding et cellspacing me laisse perplexe. Effectivement celles-ci permettent de préciser le padding et le spacing pour toutes les cellules. Or il n'y a pas, à ma connaissance, d'équivalent en CSS. Il faudrait le préciser pour chaque TD. C'est lourd. Plus important je n'ai vu aucune remarque sur des méta-balises. C'est-à-dire la possibilité pour l'utilisateur de définir des balises d'une ou d'un groupe de balises. Cela permettrait souvent une économie d'écriture et une augmentation de la rapidité d'analyse des pages par les navigateurs. Il existe aujourd'hui un mécanisme de ce type dans les récents microprocesseurs Intel.

Quoiqu'il en soit je vais voir comment modifier mes tableaux pour supprimer les expressions cellpadding et cellspacing. C'est gênant car j'avais trouvé un truc pour faire des bordures assez simplement et sans utiliser border. C'est un vieux truc mais qui restait valable sous XHTML 1 donc je n'ai jamais eu de raison de le réviser. Maintenant c'est le cas.

Bogues en cascade : Signe que ça avance, après avoir fait un gros travail de fond, puis la gestion des profils et la messagerie, je suis retourné aux scripts d'administration en ligne d'Onversity version Luxor. En vérifiant les capacités de modification des éditeurs, je suis tombé sur une succession de bogues incroyables Cela m'a promené de scripts en scripts et de module en module. Certains de ces bogues étaient de simples problèmes de codage, mais d'autres étaient

plus sévères. Je citerais par exemple le fait que la gestion des cookies de la nouvelle version entrerait en conflit avec les cookies de la version actuelle. Ainsi lors du passage à la nouvelle version, les utilisateurs qui avaient un cookie ancienne version auraient été dans l'incapacité de s'identifier sur la nouvelle version car ils seraient entrés dans une boucle de bogues. Une autre série de bogues concernaient la fonction de modification d'enregistrement d'Hydrogène 2. A cause de la variabilité de chaque enregistrement, cette fonction est devenue un noeud de programmation du nouveau moteur de base de données. Elle a été sérieusement assainie, mais vu sa taille et sa complexité et son jeune âge je ne doute pas que je devrais y revenir. Par expérience je sais que résoudre de tels bogues permet de faire de grands pas.

Samedi 12 avril

Mes nouveaux joujoux : Je dispose d'outils plus performants pour le projet Luxor que pour les précédentes versions. J'utilise par exemple intensivement les outils de débogage de code Xhtml, Css de FireBug. De même que Cordial, un correcteur grammatical. Actuellement j'utilise beaucoup ces deux outils pour la mise en conformité des textes existants.

Mise en conformité des textes existants : En premier lieu il s'agit de modifier le code html inclus dans des textes qui remontent jusqu'à 1998. Je ne sais pas si je pourrais le faire pour les articles et les actualités, leur nombre étant très important. Voici quelques modifications.

- Suppression des
.
- Utilisation intensive des commandes <p>
- Utilisation du code Css margin pour éviter l'utilisation de
 ou de <p>
- Ajout systématique des codes de fermeture de balise tel que </il>
- Remplacement des options html de balise par des codes Css
- Suppression de l'option html align="center" (dans les balises Div par exemple)
- ...

Pour tester cela j'ai du modifier les scripts pour qu'ils soient eux-mêmes conformes au Xhtml et au CSS. Tout cela représente un énorme travail.

Pour le texte lui-même, il y a des changements :

- Utilisation systématique des majuscules avec Accent.
- Remplacement des apostrophes qui étaient utilisées comme des guillemets par des guillemets
- Vérification orthographique et grammaticale avec Cordial
- ...

Lundi 7 avril

Avatars prédéfinis : Il y a actuellement 46 avatars prédéfinis, et je termine l'intégration de 49 nouveaux. 13 concernent des personnages historiques comme Léonard de Vinci ou Ramanujan, 3 sont symboliques comme la spirale logarithmique ou les anneaux de Borromée, les autres étant d'un nouveau type. Ils concernent les jeux et les films de science-fiction.

L'objectif premier des avatars prédéfinis est pédagogique, c'est une introduction rapide à un concept ou à un personnage. L'idée d'introduire les jeux et les films de science-fiction est de les utiliser pour distiller de la connaissance scientifique. Dans un principe similaire, il existe des auteurs qui expliquent les sciences en explorant l'univers de Star Wars ou de Superman.

Dimanche 6 avril

Sécurité encore et encore : J'ai modifié les fonctions de cryptage/décryptage. J'ai aussi validé une fonction de test automatique de cryptage / décryptage. L'objectif est de diminuer significativement les risques par une attaque de force brute ou par analyse. Cela a impliqué une augmentation de la taille du texte crypté.

Les emails et les mots de passe seront cryptés dans les bases de données. J'ai récemment eu des doutes concernant la confiance qu'on peut avoir envers des personnes travaillant pour un hébergeur. Cela modifie la taille de ces champs et donc tous les scripts déjà écrits doivent à nouveau être vérifiés. Encore et encore ...

Une procédure de sécurité va être mise en place dans chaque script. Son principe est de tester si l'utilisateur change d'adresse d'IP. Si c'est le cas, on lui demande de ressaisir son mot de passe. Toutefois cette fonction fera partie des options du fichier d'initialisation et sera désactivée par défaut. Je me réserve la possibilité de l'activer si je sens un message d'attaque sur les comptes d'Onversity. Effectivement si une attaque de force brute réussit, le faussaire ne dispose pas du mot de passe, il ne disposera pas non plus de la même adresse IP. Devoir ressaisir le mot de passe sera impossible. Je réfléchis encore à cette procédure car je n'en ai pas encore vu tous les tenants et aboutissants. Il est tout à fait possible qu'au final elle ne serve à rien.

Malheureusement tout n'a pas encore été pensé en terme de procédure de sécurité. Par exemple je ne dispose pas d'un mécanisme qui modifie automatiquement l'algorithme de cryptage et oblige tous les utilisateurs à ressaisir leur mot de passe.

Il manque aussi un test d'activité des scripts. L'idée est de pouvoir tester si un script est appelé plusieurs centaines de fois ou plus par la même personne dans un laps de temps donné. On peut aussi le voir comme une procédure apparentée à une technologie anti-flood. Dans ce cas il suffirait d'obliger qu'un certain temps s'écoule entre deux consultations d'un script. Cela empêcherait une attaque massive pour casser un code. Vous ne pouvez pas vous imaginer le temps que sont près à perdre certains pour vous casser votre site internet.

Bon ben j'ai du travail sur les scripts que j'ai déjà fait et refait.

Mercredi 2 avril

Fichier d'initialisation : Il existe dans la version actuelle un fichier d'initialisation, mais voilà à quoi il ressemble

```
000000000000  
000000000000
```

voici à quoi ressemble la nouvelle version

```
Forum_read=1  
Forum_write=1  
Mobile=0  
Cookie=1  
Privat_message=1
```

A travers cette différence je veux vous montrer sans en faire des pages l'importante évolution dans la conception et le fonctionnement du futur site Onversity. Donc allons plus loin dans les explications vous verrez c'est intéressant.

L'idée derrière l'ancien fichier d'initialisation est de pouvoir activer/désactiver des sections du site. La première ligne correspond aux sections principales, la deuxième aux sections secondaires. Dans la nouvelle version se sont des fonctionnalités et plus particulièrement celles servant la sécurité ou le bon fonctionnement du site.

Forum_read= Permet de consulter en lecture les forums

Forum_write= Permet d'écrire dans les forums

Mobile=Active la vérification de changement d'IP de l'utilisateur. Si c'est le cas le cookie est désactivé et l'utilisateur doit s'identifier à chaque changement d'IP

Cookie=1 Active la reconnaissance par cookie. Dans le cas contraire il faudra ressaisir son mot de passe à chaque session. C'est une mesure de sécurité. L'identification par cookie génère une faiblesse potentielle

Privat_message=1 Activation ou désactivation des messages privés. Une mesure de prévention en cas d'un problème d'utilisation abusive par nuisibles.

Voyons voir à présent les différences dans le code. La version actuelle

```
int lit_droit()
{
    int    hd;

    char   *chp,
           *fic;

    chp = (char *) malloc(256);
    fic = (char *) malloc(64);

    strcpy(fic,datas_path);
    strcat(fic,"droits/droits.txt");
    hd = use_ref(fic);
    read(hd,ledroit,sizeof(ledroit));
    close(hd);

    free(chp);
    free(fic);

    return 0;
}
```

La version Luxor

```
int get_init_var_subroutine(int default_init_var, char *sub_string)
{
    int    ret;
    char   *chp;

    if((chp = strstr(init_string,sub_string)) != NULL)
    {
        chp += strlen(sub_string);
        if(*chp == '=')
        {
            if((*chp >= '0') && (*chp < '9'))
            {
                ret = (unsigned char) *chp - 48; // les valeurs possible sont
                0/1 pour fermé/ouvert - default : 1
            }
            else
                ret = default_init_var;
        }
        else
            ret = default_init_var;
    }
    else
        ret = default_init_var;

    return ret;
}

int get_init_var(int init_var)
{
    int    ret;
    switch(init_var)
```

```

    {
        case Forum_read      : ret = get_init_var_subroutine(1,"Forum_read");
                               break;

        case Forum_write: ret = get_init_var_subroutine(0,"Forum_write");
                               break;

        case Mobile          : ret = get_init_var_subroutine(0,"Mobile");
                               break;

        case Cookie          : ret = get_init_var_subroutine(1,"Cookie");
                               break;

        case Privat_message  : ret = get_init_var_subroutine(1,"Privat_message");
                               break;

        default              : ret = 0;
                               break;
    }
    return ret;
}

int read_init_file()
{
    int    ret;
    char   *init_file_name = (char *) malloc(64);

    strcpy(init_file_name,datas_path);
    strcat(init_file_name,"init/onversity.ini");
    if((ret = use_ref(init_file_name)) >= 0)
    {
        if(read(ret,init_string,512) < 0)
            ret = -2; // lecture du fichier d'initialisation impossible
        close(ret);
    }
    else
        ret = -1; // ouverture fichier impossible

    free(init_file_name);

    return ret;
}

```

La version Luxor contrôle les opérations sur le fichier d'initialisation. Elle n'est pas dépendante de la position des variables mais de leur nom seulement. Le fichier d'initialisation est plus clair et facile à modifier. La version Luxor gère aussi des paramètres par défaut en cas d'impossibilité de lecture du fichier d'initialisation. La version actuelle ne le fait pas. On accède par des fonctions aux paramètres permettant d'éviter l'utilisation d'une variable générale déclarée en externe.

Les changements apportés dans la manière de gérer les variables d'initialisation sont significatifs de bons nombres de changements sur Onversity. Beaucoup de fonctions disposent à présent de nombreux contrôles d'erreur absents précédemment. Des situations nouvelles sont envisagées comme ici les paramètres par défaut. La manière dont doivent être codés en langage

C les scripts est améliorée. Facilité d'utilisation, de maintenance et de lisibilité sont des lignes directrices dans l'écriture. Onversity 1999 est bâti en terme de performance avec un Pentium II en tête. Onversity 2008 avec un processeur Nehalem 8 cores de 4 GHz. On peut donc sacrifier en performance certaines fonctions actuelles.

Vendredi 28 Mars

Caractères accentués et spéciaux : Exploiter des caractères accentués avec les CGI c'est une plaie. Il y a d'abord le passage de paramètre d'un script à l'autre, ensuite il y a les particularités de certains navigateurs comme IE qui, dans le passage de caractères accentués à une fonction javascript, font n'importe quoi ce qui oblige à faire des fonctions de codage / décodage spéciales pour tromper IE tout en étant compatibles avec les autres et cohérentes avec les principes de codage. Pour ce dernier point, j'ai pu récupérer l'expérience de la version actuelle. Il y a aussi les cookies. Eux aussi ont eu le droit à une paire de fonctions codage / décodage pour pouvoir gérer les caractères spéciaux.

Toujours est-il que je dispose aujourd'hui d'une librairie pour traiter la majorité des cas. On verra si elle est suffisante dans les mois à venir. Mais il est clair qu'une gestion cohérente des accents et des caractères spéciaux aurait du être menée depuis longtemps.

Samedi 22 Mars

Fonction de cryptage corsée : Je ne dirais évidemment rien sur l'algorithme utilisé. En matière de sécurité moins on en dit, plus on se protège. La procédure de cryptage, bien que généraliste à tout type de message, concerne avant tout le cookie d'identification qui évite de saisir son mot de passe à chaque venue. Tout le problème réside à faire en sorte qu'on ne puisse pas en fabriquer un (le cookie) pour se faire passer pour quelqu'un. Le cryptage est non seulement plus sophistiqué, mais il change à chaque création du cookie pour un pseudo donné sans répétition, même au bout de 1000 connexions et la manière dont cela change est différente pour chaque pseudo.

Marre de l'informatique américaine : J'ai eu la désagréable surprise, lors de test systématique des algorithmes de cryptage, de voir que l'écriture et la lecture de cookies posait des problèmes pour certains caractères ayant un code ASCII > 127. Je ne sais pas si cela est du au protocole HTTP (ici IIS de Microsoft) ou aux navigateurs, mais c'est un reliquat de l'informatique américaine qui se fout des accents et des pays qui les utilisent. Combien y a-t-il de programmes européens qui fonctionnent actuellement et qui ne peuvent enregistrer les accents des noms et prénoms de nos compatriotes européens ? Trop. On est en 2008 et ça dure toujours.

Bref pour palier à cela, les valeurs des cookies sont codés en chiffres de 0 à 9 et convertis dans les caractères correspondants. Un codage possible pour 'jfmaquine' est 09346E7595F1F17B98A512. Donc l'écriture d'un cookie implique un cryptage, un codage, un décodage et un décryptage.

Lundi 17 Mars

Vers le 64 bits à marche forcée : J'ai appris au détour d'une discussion avec mon hébergeur que le serveur de scripts passera d'ici quelques semaines à mois en 64 bits. Il n'y aura pas de période intermédiaire d'adaptation. Les conséquences sont doubles. En premier la nouvelle version devra fonctionner directement en 64 bits. J'ai préparé le code pour cela, mais je ne dispose pas encore d'une station de travail 64 bits car rien n'indiquait que je devrais passer obligatoirement tout Onversity en 64 bits cette année. En second, il faut prier pour que le code 32 bits actuel fonctionne sous 64 bits. Je vais tout recompiler sans options d'optimisation sauf -O2. Le code de la version actuelle n'est pas compatible avec le 64 bits. Effectivement Linux 64 bits est basé sur la norme LP64 qui implique que les longs sont en 64 bits. Or j'utilise dans le code actuel des variables déclarées long devant fonctionner exclusivement en 32 bits. On a pu déjà tester que GCC 64 bits permet de compiler des scripts CGI Onversity en 32 bits qui semblent fonctionner. On utilise pour cela l'option -m32.

Luxor est conçu dès le début pour pouvoir être compilé et fonctionner indifféremment en mode 32 bits ou 64 bits sous Linux ou Windows. En utilisant exclusivement le type int est en faisant attention au sizeof(char *) ça devrait fonctionner. Onversity n'a pas besoin d'un adressage 64 bits de variable 64 bits. C'est-à-dire même lorsque le processeur utilise des registres 64 bits seuls 32 bits de ces registres sont utiles. Dans le cas de fonctions qui utilisent des registres 64 bits, deux situations se présentent. Soit Onversity fournit une donnée 32 bits pour une variable 64 bits de la nouvelle fonction. Dans ce cas le casting est automatique et ne produit aucun warning à la compilation, soit on transfère dans un int 32 bits une variable 64 bits. Dans ce cas il faut faire un casting manuel

exemple : `len = (int) strlen("texte");`

Dans le cas d'Onversity, cela ne pose aucun problème car tout est conçu pour fonctionner en 32 bits. Il ne peut donc pas y avoir de production de variable nécessitant plus de 32 bits à moins qu'un rayonnement cosmique génère un bogue spécial. On fera avec :D.

Lundi 10 Mars

Liquider les Goa'uld : J'avance lentement sur la gestion des profils et des messages privés car je dois affronter une armée de Goa'uld qui parasite la nouvelle version. Sale bête. Cela va du choix de la forme graphique des sous menus secondaires à la manière de programmer des fonctions basiques gérant le séparateur entre les heures et les minutes. Tout doit être réfléchi pour être coordonné avec les fonctions existantes et leur mécanisme et les futures fonctions. Parfois pour me donner l'impression d'avancer, je fais l'impasse sur certaines modifications. Mais sur lesquelles je reviens un jour ou l'autre comme la gestion des messages d'erreur d'Hydrogène 2 qui nécessite encore de la patine.

Faire des interventions quand nécessaire : Je ne ferais plus d'interventions avant d'avoir

terminer des étapes. Donc ne vous inquiétez pas trop si vous n'avez plus de nouvelles.

Problème de sécurité : La future version gère de nouveaux types d'erreurs. Certaines d'entre-elles peuvent permettre à quelqu'un de mal intentionné de saturer les fichiers de log d'erreurs. La solution la plus simple est, pour ces erreurs, de ne pas les inscrire dans le fichier de log tout en les signalant à l'utilisateur. Mais la qualité du retour d'information est médiocre. L'autre solution est beaucoup plus sophistiquée. Elle consiste à mettre en place une détection de type Flood comme sur les forums. Toutefois les forums sont structurés en base de données, les fichiers de log sont de simples fichiers texte. Dois-je transformer la gestion des erreurs en base de données ? Je n'aime pas cette solution et pourtant. Il faudra que je prenne une décision mais pour l'instant je vais laisser mûrir.

Je suis confronté en permanence à des tonnes de choix comme celui-là et souvent je n'ai pas les réponses immédiatement.

Samedi 23 Février

Retour au travail : Après quelques semaines de vacances je suis de retour sur la programmation de Luxor. En fait j'y suis depuis 2 semaines à nouveau. La gestion des profils est faite ce qui inclut les scripts d'inscription, d'identification, de rappel de mot de passe, de gestion des paramètres du profil, le nombre de paramètres ayant doublé.

Message privé : J'ai presque terminé la gestion des messages privés. C'est une fonctionnalité inédite sur Onversity. Cela permettra aux membres d'Onversity d'échanger des informations en toute sécurité et sans risque d'être remarqué par un moteur de recherche. Cela a aussi l'avantage de pouvoir dialoguer avec des auteurs d'Onversity sans devoir laisser un e-mail utilisable par les spammeurs.

Skin se dit parure en français : J'ai été très inspiré la semaine dernière et je me suis attaqué aux maquettes de skins. C'est-à-dire à des versions d'Onversity habillées de couleurs différentes et d'un motif différent pour le large bandeau du haut. Mais la présentation, l'organisation ne change pas pour chaque skin. Cela permet une grande quantité de skins et même des concours de skins. Je ne montrerais rien et ceux qui ont déjà vu quelques maquettes seront agréablement surpris des améliorations. Les thèmes déjà faits sont :

- Classique (passe partout)
- Quarks (une variation plus d'jeun du skin classique)
- Effet Cerenkov (Magnifique bleu)
- Radiation (Beaucoup de personnalité, l'oserez-vous ?)
- Matière noire (une petite pensée à ceux qui aiment les fonds dark)
- Géante rouge (un rouge sang associé à une couleur surprenante)
- Lave (Volcanique)
- Original (reprend les couleurs et le fond actuels qui sont celles d'Onversity depuis plus de 5 ans maintenant).

Avec les skins j'en ai profité pour retravailler les codes CSS et Xhtml. En fait c'était obligatoire pour pouvoir être pleinement programmable. Aujourd'hui en chargeant le bon fichier CSS, le skin change. Le code a été aussi débogué grâce à HTML Validator et optimisé. Par exemple à

chaque fois que cela a été possible les codes
 ont été supprimés pour être remplacés par des codes CSS Margin. Un contrôle de l'utilité de chaque box Div a été fait afin d'en diminuer leur nombre ou leur imbrication.

Et la suite ? : Reprendre les scripts de profil pour les adapter aux skins paramétrables. De même pour les messages privés en les terminant. Ensuite dans le désordre il y aura les forums, la page d'accueil et certaines fonctions d'administration. Cela devrait déjà représenter deux mois de travail.

Mercredi 2 janvier

On resigne pour une année : Il y a encore beaucoup de travail sur la planche donc on resigne pour une année. Le premier objectif pour ce début de mois de janvier est de terminer la gestion des profils et tout ce qui l'entoure. Ce 'ce' est assez important puisqu'il inclut tant la gestion des cookies que la présentation générale de chaque page et la gestion des fichiers y afférents que la gestion des skin. Cela signifie qu'avec la gestion des profils vient la gestion de toutes les parties communes à tous les scripts d'Onversity. A cela il faut ajouter qu'il ne s'agit pas seulement des scripts de profil utilisateur mais aussi d'administration et les éléments communs d'administration. Il faudra donc un certain temps avant que je passe à l'étape suivante à savoir la gestion des messages privés. Ceux-ci sont importants car ils permettent à des internautes de communiquer entre eux de manière totalement anonyme.

De la continuité : Je continuerais à vous informer régulièrement des avancées ou des problèmes rencontrés.

Lundi 24 décembre

Halte, papier : Les différences dans la manière dont sont traités les paramètres d'un script sont de plus en plus marquées entre la version actuelle d'Onversity et la future version. Il y a plus de tests où les données doivent montrer patte blanche. Cela n'est pas seulement pour détecter des tentatives d'utilisations abusives, mais aussi pour détecter sur quel point précis l'utilisateur a commis une erreur et l'aider de manière plus précise.

Devenir plus mature : La future version d'Onversity sera plus mature. Par exemple actuellement quand on s'inscrit ou qu'on modifie son profil, on revient directement sur la page d'accueil. Lorsqu'on valide un formulaire, il est intéressant de savoir clairement si la saisie a bien été prise en compte. Dans la future version, des scripts sont donc ajoutés pour informer l'utilisateur que tout s'est bien passé. Si ça se passe mal, on est réorienté vers le gestionnaire d'erreurs. Ça ça existe déjà, avec la différence pour la future version que chaque erreur que rencontre un utilisateur est stipulée dans un fichier de log des erreurs.

Vendredi 14 décembre

Une marée de détails : Quand j'explique ce que je fais, généralement je vais vers les grandes lignes, quoiqu'avec vous, je suis un peu plus bavard. Mais en fait dans mon travail il y a une foule de détails qui viennent parasiter les grandes lignes. Actuellement j'essaie de boucler la gestion des profils tant du côté des scripts d'administration que du côté utilisateur. Du côté utilisateur comme ce sont les premiers vrais scripts, il y a une foule de détails à régler. Ca va jusqu'à la forme graphique du bouton de validation et de sa couleur. Si on veut quelque chose qui ait de la gueule et cohérent on ne coupe pas à ce genre de chose.

Des cookies de Noël : Dans les détails il y a évidemment les fonctions générales (utile à de nombreux scripts si ce n'est à tous) qui ne sont pas encore implémentées ou mal. Il en va ainsi des cookies. La précédente version d'Onversity dispose de cookies, mais assez primaires. La nouvelle version sera à même de gérer

- Des cookies multiples (nécessaires pour gérer au moins un premier cookie pour l'administration et un second pour l'utilisation)
- Effacement des cookies (autoriser l'utilisateur à supprimer toute trace à partir d'Onversity)
- Variabilité de la date d'expiration (permettre à l'internaute de choisir son niveau de sécurité)
- Cryptage des cookies (ne pas permettre à une personne mal intentionnée d'utiliser la structure des cookies pour les imiter facilement et se faire passer pour quelqu'un d'autre)

De plus ces nouvelles fonctions en utilisent de nouvelles comme 'buffer_overrun' qui a en charge de tester les entrées provenant de l'environnement HTTP. Un test de buffer overrun existait déjà pour les précédentes versions d'Onversity mais seulement pour les paramètres passés entre scripts..

Je vais vous donner un truc. L'effacement de cookies s'effectue en le réécrivant et en lui donnant une date d'expiration inférieure à la date du jour. C'est ensuite le navigateur de l'internaute qui se charge d'effacer physiquement le cookie sur sa machine. Ceci est en cohérence avec le fait qu'un site internet ne doit pas pouvoir accéder directement au système d'exploitation de la machine de l'internaute par les fonctions de base du navigateur.

Vendredi 7 décembre

Les affaires reprennent : Houlala, comme le temps passe vite. Bon ben la dernière fonction importante d'Hydrogène 2 est faite avec quelques bogues en moins pour les autres fonctions. J'ai donc repris là où je m'étais arrêté il y a deux semaines à savoir la gestion des profils du côté administration. Ils devraient être terminés demain. Ensuite je fais l'implantation de la gestion de profils pour les utilisateurs. Très similaire à ce qui existe déjà mais avec des options en plus. Aller je vous en donne une : Navigateur compatible CSS3 (Oui / Non). Je vous laisse imaginer la fonction qui implique les CSS3.

Destination Forums : L'objectif, dans les semaines qui viennent, c'est de faire les forums associés à une fonction manquante depuis le début sur Onversity. Cette fonction pourrait permettre de ne plus devoir donner les adresses e-mail du webmaster sur le site.

Jeudi 22 Novembre

Everest en vue : Je pensais pouvoir m'en passer, le temps que je me remette de mes émotions d'avoir si bien avancé sur Hydrogène 2, mais la réalité m'a rattrapé et le développement de la nouvelle version d'Onversity exige sa mise en place immédiate. Conséquence je retourne au taf sur Hydrogène 2 et quel taf !

La fonction en cause est celle qui se charge de remplacer des champs dans un enregistrement. Dans les précédentes versions, deux cas se présentaient. Soit le changement de certains champs impliquait aussi des champs servant à l'indexation de l'enregistrement soit ce changement ne l'impliquait pas. Dans le premier cas il fallait re-trier à la volée les fichiers d'index. Avec Hydrogène 2 on a ce cas plus un autre. Il s'agit du cas où le champ à changer a une taille plus petite que la nouvelle valeur. Cela est dû à la taille variable des champs. Avec les précédentes versions, soit l'information avait la taille exigée, soit elle ne l'avait pas. Dans ce dernier cas on la tronquait en envoyant un message dans le fichier de log. Avec la nouvelle version on peut avoir un changement de taille qui implique un dépassement de capacité sans que cela soit une erreur. Dans ce cas l'enregistrement doit être réécrit à une autre position et les champs d'index modifiés.

Prenons l'exemple d'un forum. Un user écrit un court texte. Puis après validation se ravise car il trouve que son texte manque d'explication. Le champ texte n'est pas un champ d'index, mais sa taille change et ne peut plus être contenu dans la place allouée par l'enregistrement actuel. Il faut le réécrire comme si c'était un nouvel enregistrement.

Evidemment c'est une procédure lourde surtout pour l'algorithme d'Hydrogène. L'un des moyens retenus pour éviter cette situation est de toujours allouer plus de place que n'en a besoin réellement un enregistrement. Une sorte de marge de sécurité. Il existe donc au sein d'Hydrogène 2 un algorithme qui a en charge de calculer la taille la plus intelligente pour un enregistrement. Intelligent en ce sens qu'il doit répondre à la fois à la contrainte de taille minimale de l'enregistrement et à la contrainte d'éviter au mieux les situations impliquant la réécriture de l'enregistrement

Samedi 17 Novembre

Un vent de folie : Ici, souffle un vent de folie avec l'arrivée de la version bêta d'Hydrogène 2. Les jeunes filles sont en fleur et les mâles poussent leurs cris. Dans cette ambiance festive, les forums seront le prochain objectif avant même de réadapter les scripts déjà faits à la version 2.0 d'Hydrogène. Qui dit forums dit gestion des profils et des utilisateurs de même qu'une fonction

de communication manquante dans les forums d'Onversity.

Vendredi 16 Novembre

Ses premiers pas : Les premiers tests de performances sont tombés. Ceci signifie d'une part que le moteur Hydrogène 2.0 est capable de faire des insertions et des recherches. D'autre part, que ses performances sont excellentes. Le code est plus lourd pour chaque fonction d'Hydrogène 2.0 comparé à la version 1.5 ou 1.0. Mais ce code plus long à exécuter est compensé par des accès aux bases plus courts car occupant moins de place.

Mais la partie n'est pas finie et d'autres tests sont à réaliser avant de passer en version Bêta. Toutefois vu comme se comporte Hydrogène 2.0 pour l'instant, tout me laisse à penser que la version alpha tombera ce WE pour laisser place à la version Bêta. L'un des tests sera de contrôler enregistrement par enregistrement que les tris s'effectuent bien. Un contrôle des pointeurs devra aussi être réalisé pour vérifier s'ils se comportent comme ils le laissent penser. Si tout va bien, lundi je recommence à coder la nouvelle version d'Onversity avec le nouveau moteur.

Samedi 10 Novembre

Version Alpha : Les principales fonctions d'Hydrogène 2.0 sont implémentées. Je commencerais les tests de fonctionnement en situation réelle cette semaine. Ce qui devrait nous amener vers une première version bêta d'ici deux semaines au plus. Une fois que je serais assez sûr de la version bêta, je commencerais à modifier tous les scripts déjà faits pour la nouvelle version d'Onversity.

A la question pratique : pourquoi a-t-on besoin d'Hydrogène 2.0, la réponse est multiple, mais deux suffiront. Onversity dispose d'un compte limité en taille, diminuer la taille des bases permettra de gagner plusieurs dizaines de méga-octets. La gestion des forums et la structure des bases de données vont s'en trouver grandement simplifiées.

L'algorithme de test que j'utilise pour valider les versions d'Hydrogène est aussi un algorithme d'évaluation des performances. Au début je pensais qu'Hydrogène 2.0 serait nettement plus lent qu'Hydrogène 1.5, mais certaines réalités que j'ai découvertes en programmant le code d'Hydrogène 2.0 me laissent à penser que la différence pourrait ne pas être aussi importante que prévu. Voire on pourrait avoir une surprise. La réponse d'ici une semaine j'espère.

Lundi 5 Novembre

La clef des champs : aujourd'hui je devais adapter une fonction nommée "Ecrit_champ" pour

Hydrogène 2. La version précédente est simple et ne prend que quelques lignes. La nouvelle version est apparue beaucoup plus complexe que je ne m'y attendais. Cette fonction a pour but d'écrire un champ dans un enregistrement à la bonne position. Avec la variabilité des tailles de chaque champ, on a aussi la variabilité de leur position. Et bien cette variabilité entraîne beaucoup plus de contraintes que prévu et donc une difficulté d'écriture. Une autre différence avec les précédentes versions d'Hydrogène est que "Ecrit_champ" devient centrale pour l'insertion de données dans la base, alors qu'auparavant cette fonction était secondaire. On pouvait s'en passer.

C'est presque officiel cette fonction risque de devenir ma tête de turc. Je ne l'aime pas. J'en ai une aussi pour les anciennes versions, il s'agit de la fonction Search. On verra avec le temps si nos relations s'apaisent ou si je dois la fouetter de temps en temps :D.

Dimanche 4 Novembre

Ses premiers pas : J'ai pu compiler et lancer une initialisation de librairie suivie d'ouverture de base en écriture. Première correction de bogue aussi. Comme dirait Frank "it's alive !". Je continuerais à ajouter les fonctions essentielles pour pouvoir faire les premières écritures et lectures. C'est au début que le boulot est le plus gros, il faut mettre au point tout un ensemble d'indicateurs qui sont bien plus nombreux sur Hydrogène 2 sans compter une complexité légèrement accrue au niveau des pointeurs. Une fois les principaux mécanismes en place, il me suffira presque de faire un copier / coller entre Hydrogène 1.5 et 2.0 et de modifier les quelques différences.

Une chose dont je n'avais pas pleinement pris conscience est que le nombre d'enregistrements maximum d'une base à une autre est variable. Le nombre minimum d'enregistrement est 2, le maximum 2^{32} . Dans ce dernier cas, on est en mode compatibilité avec les précédents Hydrogène.

Mardi 30 Octobre

Hydro sans gêne : Voilà, ça fait 6 ans que j'en ai eu l'idée sur une plage du pays basque. L'écriture d'Hydrogène 2.0 est en route. Ceci terminera un ensemble de choix technologiques que j'avais prévu d'intégrer à Onversity depuis plusieurs années. J'espère que d'ici 1 à 3 semaines je pourrais vous dire si cette partie se passe bien ou pas. La gestion des pointeurs y est plus délicate.

Jeudi 04 Octobre

Ne jamais faire confiance à un compilateur : Je prépare un petit article sur les performances comparées de YOUM 2 et 3 sur processeurs P4c Northwood et Core 2 Duo et des compilateurs Intel version 8, 9.1 et 10 et Borland BC5. Ce que ce tableau m'a indiqué est que BC5 sous Core 2 duo avec YOUM3 est complètement à côté de la plaque en fournissant une différence de performance de 40% avec YOUM2 contre 20% dans les autres situations. Comme pour les

derniers tests de différence de performance YOUM2 et 3 je me basais sur ce compilateur j'ai plutôt eu une bonne surprise, puisque le delta est moins important que prévu.

Cela montre aussi que les contres performances des compilateurs pour un code donné existent. Elles sont peu fréquentes mais pas exceptionnelles.

Lundi 01 Octobre

Une révolution en marche : Le passage de la technologie mécano-magnétique des disques durs actuels à une technologie purement électronique va bouleverser l'informatique. Notre conception du stockage va devoir changer. Nous allons assister à une fusion des mémoires de stockage et des mémoires de travail. Mais avant cela des changements vont se faire sentir. Par exemple tous les moteurs de base de données sont conçus pour offrir un temps d'accès minimum. Or avec une multiplication par 100 des performances en accès certaines conceptions devront être revues. La nécessité de gérer une mémoire cache en RAM pour les moteurs de base de données se posera à un moment.

Cette fusion pourrait aller bien plus loin avec l'avènement de liaison optique entre processeur. On pourrait voir apparaître une unique mémoire centralisant les données pour l'ensemble des processeurs (CPU, GPU, ...). C'est l'organisation complète du flux des données qui pourrait subir un bouleversement.

Lire la suite

Dimanche 12 Août

En route Simone ! : La semaine va être courte pour moi mais je commence dès aujourd'hui les modifications d'Hydrogène 1.5 pour passer en Hydrogène 2.0. Je devrais être à même de faire des tests dès la fin de la semaine prochaine.

Sur les fondements des moteurs de bases de données (suite) : Il existe 4 méthodes principales de recherches. Séquentielle, dichotomique, arborescence et hachage. La méthode séquentielle n'est intéressante qu'en dessous de 8 éléments. La méthode de hachage est la plus rapide et fournit un temps d'accès assez stable de l'ordre de 2 à 4 éléments. Les arbres équilibrés offrent la même complexité que la dichotomie à savoir $O(n \cdot \ln(n))$ mais sans un problème épineux d'insertion. Grosso modo voila un résumé assez libre de ce que vous pourrez trouver dans tout livre d'algorithmique.

Dimanche 12 Août

Les contraintes des enregistrements concaténés : Il est nécessaire de connaître le nombre d'enregistrements d'une base de données. Dans Hydrogène 1 ceci est fourni implicitement par la taille du fichier maître par la formule $\text{taille du fichier en octet} / \text{Taille d'un enregistrement en octet}$. Cela fonctionne puisque chaque enregistrement a la même taille. Avec Hydrogène 2 cela ne fonctionne plus. On pourrait stocker cette information dans un fichier à part ou dans le fichier maître, mais il est préférable d'exploiter l'idée d'information implicite. Ainsi la taille de chaque enregistrement restant fixe, il suffit de connaître la taille d'un des index pour avoir le nombre d'enregistrements. Les contraintes sont donc que seuls les fichiers d'index peuvent informer sur le nombre d'enregistrements et que l'ouverture des fichiers doit impérativement se faire par le ou les index en premier.

Une autre contrainte est que le dernier enregistrement d'un index doit représenter le nombre de bloc total du fichier maître afin de pouvoir connaître la taille du dernier enregistrement. On en revient à la formule $E_{n+1} - E_n = m$, où m la taille en bloc de l'enregistrement E_n .

L'écriture des champs d'un enregistrement doit se faire dans l'ordre de déclaration des champs puisque la position de chaque champ n'est pas définie à l'avance. Ce type d'écriture se produit à la création d'un nouvel enregistrement avant insertion dans la base. Avec Hydrogène 1 l'ordre n'a pas d'importance.

La lecture d'un enregistrement implique la création d'un tableau de pointeur vers les champs pour chaque enregistrement. L'affectation de ces pointeurs implique de faire une lecture de chaque caractère par une boucle itérative. Cela consomme un peu de ressource processeur. Hydrogène 1 n'a pas cette nécessité. Cela consomme aussi des ressources mémoire supplémentaires. Toutefois ces ressources peuvent être compensées, la concaténation permet une économie de place supérieure ou égale.

Jeudi 9 Août

Sur les fondements des moteurs de base de données : Il existe fondamentalement deux types de structure dans une base de données pour gérer les données. L'une stocke les données, on parle parfois de fichiers maîtres, l'autre gère les index. Un moteur de bases de données est entièrement déterminé (comme disent les mathématiciens) par sa méthode de recherche. Dans le cas de la dichotomie, la structure des index implique de disposer d'au moins un champ contenant la donnée permettant d'avoir l'adresse où sont situées toutes les informations d'un enregistrement. Dans le cas du couple fichier maître / index, on dit que l'index pointe vers l'enregistrement du maître.

Toutefois inscrire dans ce champ spécial de l'index directement l'adresse physique de l'enregistrement pose un problème. Si l'adresse est codée sur 2^{32} bits et que la taille de chaque enregistrement fait 2^{13} , vous ne pourrez disposer que de 2^{19} enregistrements. Il existe des solutions pour résoudre ce désagrément, mais elles dépendent de vos enregistrements, selon qu'ils aient une taille fixe ou variable. Fixe signifie que les enregistrements ont tous la même taille et que cette taille est fixée à l'avance et ne dépend en aucun cas du contenu. Dans ce cas il suffit d'indiquer le rang de l'enregistrement. Pour obtenir son adresse, il suffira de multiplier son rang par la taille de l'enregistrement. Dans le cas d'enregistrements variables (ou concaténés) il faut donner l'adresse. Toutefois la taille d'un enregistrement est généralement considéré comme étant constitué de bloc de 1 octet. En admettant que les blocs soient constitués de N octet avec N de type 2^n pour plus de facilité et pour l'exemple précédent alors on peut disposer de $2^{(32-13+n)}$ enregistrements. Si n vaut 4 alors on dispose de 2^{23} enregistrements contre 2^{19} . Evidemment pour des enregistrements aussi gros que 2^{13} on peut considérer des blocs de taille plus grande comme 2^8 .

On touche ici une différence fondamentale entre enregistrement de taille fixe et variable. Les fixes permettent de gérer plus d'enregistrements pour un codage donné de l'adresse.

Pour terminer sur mes petites digressions, il reste à voir comment on code l'adresse pour des enregistrements à taille variable. La première idée est d'avoir une zone (admettons de 24 bits) pour l'adresse et une autre (de 8 bits) pour la taille de l'enregistrement en bloc de 2^n octet. Mais avec la formule suivante

$E_n + m = E_{n+1}$, il n'est pas nécessaire de stocker la taille. Cette information est déjà donnée puisque $E_{n+1} - E_n = m$. (Où E_n est l'enregistrement de rang n et m la taille en bloc de 2^n octets).

L'adressage par rang (enregistrement fixe) et par adresse physique (enregistrement variable) sont de même type. Si la taille des blocs qui gèrent les enregistrements variables est égale à la taille de l'enregistrement, alors les capacités d'adressage entre les deux méthodes sont identiques.

En reprenant notre exemple, on a $2^{(32-a+b)}$ avec a = taille de l'enregistrement, b = taille du bloc. Si $b = a$ alors $2^{(32-a+b)} = 2^{32}$. Cela tient au fait qu'en affectant $b = a$, l'ordre de croissance entre chaque enregistrement consécutif vaut 1 comme dans une organisation par rang. Cette remarque est très importante car elle implique qu'un système à taille fixe n'est qu'un cas particulier d'un système à taille variable par bloc. Dans la pratique, cela signifie que Hydrogène 2 est compatible avec la structure des index d'Hydrogène 1 et 1.5. Ainsi pour faire le transfert des fichiers de l'ancienne à la nouvelle version, je pourrais utiliser Hydrogène 2 uniquement pour faire les scripts de transfert..

Hydrogène dans sa version 2.0 ne pourra pas gérer 2 milliards d'enregistrements théoriques comme précédemment mais quelques millions dans la mesure où la taille des index restera la même. Ce qui est largement suffisant

Mardi 7 Août

Hydrogène 2.0 ? : Hydrogène est le moteur de base de données maison. Son algorithme est assez rare, car en théorie il a une mauvaise complexité en insertion. Problème que j'ai su sérieusement repousser et si à cela on ajoute que dans les autres fonctions il se montre non seulement rapide mais très réactif et que la fonction d'ajout pour un site comme Onversity est ultra minoritaire alors c'est un choix parfaitement légitime.

Toutefois Hydrogène ne gère que des enregistrements à taille fixe. Cela pose deux problèmes potentiels. Le premier est l'utilisation excessive de place. Par exemple dans la nouvelle version les articles ne sont plus découpés en enregistrement représentant chacun un paragraphe, mais il n'y a plus qu'un seul enregistrement de 80 ko pour chaque article. Interview pareil. Or évidemment aucun article utilise 80 ko de texte. Certains articles ne faisant pas 10 ko, la perte de place peut aller jusqu'à 90%. Pour les forums, c'est pire la perte moyenne de place allant jusqu'à 95%. Lorsqu'il y a une variabilité assez grande de la taille des textes comme pour les forums, on utilise, avec un moteur de base de données à taille d'enregistrement fixe, des fichiers extérieurs. Ainsi pour chaque enregistrement de chaque message on stocke dans l'enregistrement le nom d'un fichier qui ne contient que le texte. Ce fichier pouvant avoir la taille qu'il veut et prendre la place exacte de ce qu'il contient. L'inconvénient est que pour afficher une page de 20 messages, il faut ouvrir 20 fichiers sans compter les fichiers de base de données.

Je dispose donc depuis plusieurs années dans mes cartons d'un algorithme qui rend Hydrogène à enregistrements variables. Évidemment tant que ça ne sera pas implémenté, la compatibilité avec Hydrogène et cette technique restera discutable, mais j'ai bon espoir. Fondamentalement rien ne change dans la manière d'utiliser Hydrogène (tant mieux), il y a même pas trop de codes à reprendre. La plus importante modification se situe dans les index. Ceux-ci contiennent actuellement les numéros d'enregistrements. Avec cette nouvelle technique, ils contiendront une adresse absolue et un nombre de bloc. Certaines lignes de code impliquant la multiplication du numéro d'enregistrement par la taille de l'enregistrement pour obtenir l'adresse seront obsolètes. La taille des nouveaux enregistrements se calcule en multipliant le nombre de blocs de l'enregistrement par la taille des blocs qui elle est fixe.

Pourquoi faire le changement maintenant ? Et subséquemment pourquoi ne pas l'avoir fait tout de suite ? J'étais surtout concentré à écrire les premières lignes de la nouvelle version pour savoir où j'allais, changer de moteur de base de données n'était pas d'actualité. En même temps, l'idée de faire évoluer Hydrogène m'est revenue au moment où je m'attaquais aux nouveaux forums, partie qui pourrait le plus bénéficier de cette nouvelle version. Il faut bien comprendre que le passage à Hydrogène 2 sera un pas important. Le faire au moment de l'écriture de la nouvelle version qui va amener tant de nouveautés est une occasion en or. Hydrogène est aussi arrivé à maturité, on peut donc passer à une modification qui est juste dans le mode d'accès aux données, pas dans la manière dont elles sont traitées une fois en mémoire car là ça sera pareil. Il faut savoir d'ailleurs qu'une importante modification, qui a vu la naissance d'Hydrogène 1.0 et faisait suite à Maqbase 5, était conçue dans l'esprit d'une implémentation future des champs concaténés.

Des performances supérieures. Ce que pourrait permettre Hydrogène 2, si la modification est effectivement viable, c'est de concaténer les enregistrements. Les performances en débit seront appréciables. Elle le seront d'autant plus qu'actuellement Onversity est en serveur mutualisé sur un cluster de serveur. Pour les forums, le gain est évident suite aux explications de gain de place de 95%. Mais pour les actualités aussi. Ainsi en moyenne on chargera moins de 50 % de données qu'actuellement. Hydrogène permettra aussi d'utiliser des algorithmes de

compression si nécessaire. Avec l'arrivée des disques SSD et des gains d'un facteur 100 sur les temps d'accès, la pression va être mise sur les débits pour les bases de données.

Hydrogène 2 sera l'algorithme quasi-ultime pour l'algorithme de dichotomie choisi. Je me rappelle les premières lignes de code que j'ai écrites pour ce moteur en 1991. A l'époque le moteur s'appelait Maqbase. Hydrogène 2 correspondra à Maqbase 6.0. Il ne me reste plus qu'à mettre en pratique la théorie des enregistrements concaténés.

Lundi 6 Août

Semaine de transition : Je profite de cette semaine pour réfléchir à la structure des bases de données des forums et donc en même temps aux fonctions. 9a sera un forum assez complet et moderne. Je réglerais aussi 2 ou 3 détails sur le travail déjà fait.

Jeudi 2 Août

Un peu de ciel bleu : Les scripts d'administration des éditos commencent à être bien débogués et intègrent bon nombre de nouveautés par rapport aux méthodes de développement que j'utilisais avec la version actuelle. La fonction de transfert des anciens éditos vers leurs nouvelles bases est faite. En fait cela nécessite le transfert de 2 bases, mais bon. Je fais aussi des fonctions pour convertir certaines données des anciennes bases. Par exemple, un convertisseur de balise HTML 3.2 en XHTML 1 genre `
` en `
`. Il me faudra aussi faire une routine de conversion des noms des auteurs. Ceux-ci sont à présent dans une base à part. Je dois donc remplacer les noms complets des bases éditos par les pseudos des auteurs. Les scripts se chargeront, à partir du pseudo, de retrouver dans la base des auteurs toutes les informations nécessaires. Je vais donc avoir besoin que les scripts d'administration des auteurs soient à niveau. Ils existent déjà, mais beaucoup de choses manquent.

Bientôt les forums : Une fois que j'aurais suffisamment avancé avec les éditos, leur transfert et les auteurs je m'attaque aux forums. Ils reprendront toutes les fonctionnalités des forums actuels et ajouteront de nouvelles fonctions. Mais surtout la structure des bases de données sera complètement revue pour une structure plus simple. Les urls seront aussi bcp plus courtes et mieux sécurisées.

Lundi 23 juillet

Ajout des fonctions de visualisation : Toujours dans les éditos il manquait évidemment la visualisation du texte saisi avant de le valider dans les bases de données pour qu'il puisse être visible par les lecteurs. Je vais prendre du temps pour cette visualisation car sa présentation sera à la base de toutes les autres présentations de texte. Les visualisation des textes sont les mêmes

du côté administratif que du côté lecteur.

Je l'aurais cette sale bête : L'augmentation des tests de sécurité a révélé un bogue ou plutôt une insuffisance dans le moteur de base de données. Hydrogène. 9a touche une fonction très importante, je marche sur des oeufs.

Lundi 16 juillet

Lamentable ! :Tous les scripts d'administration que j'avais faits ont été testés et passés en version bêta. Le problème est qu'en finalisant les éditos j'ai du combler beaucoup de fonctions manquantes et en modifier d'autres. La situation est telle que je suis bon pour reprendre chacun des scripts déjà écrits et refaire tous les tests. Donc tout ce qui a été fait avant les nouveaux scripts éditos est repassé en version alpha. Bref j'ai été lamentable.

Lundi 9 juillet

Finalisation de la gestion administrative des éditos :Juste avant les vacances j'ai commencé à bosser sur la finalisation des éditos du côté administratif. En rentrant de vacances j'ai donc continué ce travail.

Jeudi 21 juin

Les affaires reprennent :J'ai développer la saisie administrative des éditos et des résumés de livres soit une quinzaine de scripts. Le prochain point sera de faire la version alpha de la page d'accueil qui changera pas mal comparé au proto dont je dispose actuellement.. En rentrant de vacances j'espère pouvoir bloquer plusieurs jours uniquement sur le développement.

Vendredi 15 juin

La récursivité se met en branle : De la nouvelle version je suis passé à YOUM3, de YOUM3 aux dictionnaires. Les modifs dicos terminées, je suis repassé à YOUM3 et YOUM3 étant terminé je vais pouvoir repasser à la nouvelle version. En gros c'est ce qui va arriver, mais la réalité est un peu plus biscornue. Après avoir fait un premier lot de définitions, j'ai terminé YOUM3. Il fonctionne sur plus de 1000 textes et affiche plus de mots, mais n'a pas la patine de YOUM2. Ca viendra avec le temps, mais dire qu'il est terminé ne signifie pas que le code ne doit plus être touché. Mais il n'y a plus urgence. Pour les définitions j'ai décidé de continuer à en ajouter car il y a des textes sans aucun lien, je continuerais à en ajouter jusqu'à mes vacances c'est à dire une semaine. Je pars 3 à 4 semaines en vacances. Je recommencerais à travailler sur la nouvelle version cette semaine tout en ajoutant encore une dizaine de définitions. Mais la nouvelle version reprendra surtout à mon retour de vacances.

La solution pour YOUM3 était d'utiliser la récursivité. Je rappelle le problème. Quand on fait un séquenceur de mot tout va super bien tant qu'on séquence des mots uniques et bien tant qu'on

favorise les plus petits syntagmes. Le problème survient quand on gère de longs syntagmes et surtout quand on impose de favoriser systématiquement les syntagmes les plus longs. La solution dans ce cas n'est pas triviale. En fait pour être exact, un séquenceur seul ne nécessite pas de récursivité. C'est seulement quand on ajoute la génération de lien à des dictionnaires que l'ordre d'apparition et de traitement devient important et pose problème. C'est ce problème qui a nécessité une solution particulière qui se trouve être ici la mise en oeuvre d'une récursivité à un moment donné. YOUM3 est 1,4 fois plus lent que YOUM2. Mais si on considère que YOUM3 gère un nombre illimité de syntagmes et qu'il ne possède pas certaines des optimisations extrêmes de YOUM2 on peut considérer que c'est un résultat plus que satisfaisant.

Lorsqu'on favorise les syntagmes longs, on est obligé de 'buffériser' les mots puis selon certaines conditions vider ce buffet. C'est au moment du 'vidage' qu'on construit les différentes combinaisons de mots pour former des syntagmes qui sont ensuite comparés avec les mots contenus dans plusieurs dictionnaires. 4 actuellement pour Onversity. Remarque : Il y a un sens de lecture du buffet. Admettons qu'on ait un buffet de 4 mots. On analyse d'abord le syntagme de 4 mots, puis s'il n'est pas dans un des dicos on analyse les deux syntagmes possibles de 3 mots, soit $M1+M2+M3$ soit $M2+M3+M4$ ($M1$ = premier mot du buffet). Avec YOUM3 on analyse d'abord $M2+M3+M4$. S'il n'est pas dans un des dicos on passe à $M1+M2+M3$. S'il est dans un des dicos on génère des pointeurs pour insérer un lien dans le texte. Le problème est que dans ce cas il reste $M4$ qu'il faut analyser. Mais le sens de lecture nous interdit de créer un lien vers un dico sur $M4$ si un lien existe déjà sur la séquence $M2+M3+M4$. Les pointeurs seront déphasés. La solution consiste, au moment où on a détecté que $M1+M2+M3$ avait une correspondance dans un dico, de relancer l'analyse sur le reste des mots (ici $M4$ tout seul) avant de générer les pointeurs sur $M1+M2+M3$. Cela se fait par la récursivité.

Mercredi 25Avril

Rupture du flot séquentiel : Suite du problème que pose YOUM 3 à savoir trouver un algorithme qui teste toutes les combinaisons de mots dans une liste de 'n' mots en favorisant toujours les syntagmes les plus grands tout en ayant un code conçu pour fonctionner quelle que soit la valeur de 'n'. Je crois qu'on peut prouver qu'il n'existe pas de solution séquentielle à ce problème. C'est-à-dire que ce problème ne peut être résolu par une utilisation même sophistiquée de suites de codes et/ou de boucles.

Démonstration : Dans une suite d'itérations (une boucle), on ne peut traiter soit que des syntagmes ayant le même nombre de mots, soit des syntagmes ayant un nombre de mots croissants/décroissants en utilisant une fonction continue. Effectivement lors de la découverte d'un syntagme de 'n-1' mots au plus dans une liste de 'n' mots, il faut pouvoir analyser le syntagme le plus grand avec les 1 à 'n-1' restant. Ce faisant, on peut être amené à tester un syntagme plus petit que 'n-2' mots. Cela induit une rupture du flot séquentiel ou si on reprend le parallèle avec une fonction, cela induit une discontinuité.

Par exemple dans une liste de 4 mots. En fonction du sens de déroulement du flot d'analyse, il existe une situation où l'analyse d'un syntagme de 3 mots pour lequel on trouve une occurrence dans un dictionnaire, implique d'analyser le syntagme de 1 mot restant avant un syntagme de 2 mots et aussi avant que le syntagme de 3 mots ne soit enregistré.

La récursivité offre ici une solution possible pour gérer la rupture du flot de traitement et j'espère pouvoir bientôt la tester

Dictionnaires : Je prends un peu de temps pour remettre des définitions dans les dictionnaires. Je rappelle que toute aide sérieuse est la bienvenue et qu'écrire des définitions intéressantes prend du temps.

Mardi 24Avril

Je n'avais pas pensé à la récursivité : Les tableaux ne suffisent pas pour examiner tous les cas dans un ordre qui favorise toujours la détection des syntagmes les plus longs. En faisant ce constat je me suis aperçu que la solution pourrait venir de la récursivité. En fait cette technique m'offre un nouvel éventail de solutions auxquelles je n'avais pas accès jusqu'à présent. Il est même envisageable de penser réussir à trouver un algorithme qui puisse résoudre le problème de façon entièrement automatique et sans l'assistance d'une table. C'est en fait un joli problème comme je les aime. Enfin un casse tête qui m'intéresse.

Quelques explications : Je dispose d'algorithmes qui lisent un texte en avant et en arrière. Si vous le lisez en arrière c'est-à-dire en commençant par la fin comme le fait YOUM 3, les mots les plus à gauche d'une série de mots contiguës sont les syntagmes prioritaires. Maintenant prenons le cas d'une liste de 4 mots maximum. On cherche un syntagme de 4 mots, puis de 3 mots en partant de la droite puis de 3 mots en partant de la gauche. C'est là que ça se corse. Si un syntagme est trouvé alors il reste le mot le plus à droite à analyser. Si on ne le fait pas avant de valider le syntagme trouvé de 3 mots alors, à cause de modification de la chaîne de destination ou de pointeur pour la chaîne de destination, il ne sera plus possible de l'insérer. On pourrait se dire qu'il suffirait de tester ce mot seul avant de tester les 3 mots les plus à gauche, mais si ce mot génère un lien mais pas la liste de trois mots à gauche alors on aura perdu la possibilité de tester un syntagme éventuel des deux mots les plus à droite. Or la contrainte veut qu'on favorise dans tous les cas les syntagmes les plus longs.

Il existe une solution beaucoup plus simple dont je viens de me rendre compte c'est de faire du multi-passe. Une passe pour détecter des syntagmes de 'n' mots, puis 'n-1' jusqu'à 'n=1'. Le problème est la grande lenteur de cette solution.

Vendredi 20 Avril

La réponse à la question est ... : J'ai encore essayé une nouvelle stratégie mais c'est un échec. En fait l'analyse de ce nouvel échec m'a permis d'y voir plus clair et a priori il n'existe pas de solution 'simple' voire pas du tout pour trouver une stratégie optimale automatiquement pour analyser tous les différents syntagmes d'une liste de mots de taille finie.

Je vais donc passer par la bonne vieille méthode du tableau. Il n'est pas souhaitable de revenir à la méthode de YOUM2 qui consiste à écrire un algorithme spécifique pour une stratégie de syntagmes composés au maximum de n mots donnés. Il faut pouvoir rester hautement paramétrable. L'utilisation d'un tableau de stratégie est le meilleur compromis. Cela consiste,

dans une boucle, à dire à l'algorithme quelle stratégie adopter à chaque itération pour analyser tous les syntagmes possibles. La stratégie retenue pour des syntagmes de 5 mots est

5
0-4
4-0
0-3
3-2
2-2-1
1-2-1-0
0-1-1-0-0

Chaque chiffre représente le nombre de mots associés pour un syntagme. On commence par des syntagmes de 5 mots pour arriver à des syntagmes de 1 mot. Il faut pouvoir tester toutes les combinaisons de syntagmes de 1, 2, 3, 4, 5 mots dans une liste de 5 mots. Il faut aussi faire un tableau pour 4, 3 et 2 mots. Par exemple pour 4 c'est

4
0-3
3-0
2-2
1-2-1
0-1-1-0

Maintenant n'allez pas vous imaginer que l'utilisation d'un tableau simplifie le travail.

Jeudi 19 Avril

Les cadavres refont toujours surface : Ce matin je me suis réveillé avec une révélation. La stratégie de détection des syntagmes n'est pas bonne. Elle est meilleure que la première version de YOUM 3, mais ne prend quand même pas les syntagmes longs autant que YOUM 2.

En fait je me retrouve avec un problème que je n'ai jamais réussi à résoudre avec YOUM 2. Il faut bien comprendre que l'analyse prioritaire des syntagmes longs dans une suite de mots est un problème majeur. Faire un algorithme qui favorise les syntagmes courts voire à mot unique est finalement assez simple. En faire un qui systématiquement favorise les syntagmes longs avec l'analyse des syntagmes les plus courts est hautement complexe surtout si on met en avant le critère des performances. Rapide signifie plus de 10 millions de symboles par seconde au moins sur un seul core d'un processeur grand public.

Voilà la stratégie actuelle de YOUM 3 et celle vers laquelle il faut aller.

XXXX
XXXO
XXOO
XOOO
OXXX

OXXO
OXOO
OOXX
OOXO
OOOX

XXXX
OXXX
XXXO
OXXX
OXXO
XXOO
OOOX
OOXO
OXOO
XOOO

xxxx signifie l'analyse d'un syntagme de 4 mots alors que oxxo signifie l'analyse d'un syntagme de 2 mots centraux dans la queue des mots.

Il faut savoir que YOUM 2 lui-même n'est pas parfait, mais il est un peu meilleur que YOUM 3. L'objectif étant d'avoir un algorithme qui permette de paramétrer la longueur de la queue des mots avec un seul paramètre et que l'algorithme s'adapte automatiquement pour une analyse optimum.

La seule incertitude que j'ai est si ce problème a une solution et comment savoir s'il y en a une sans la trouver avant. Mais bon je dispose déjà d'un algorithme paramétrable c'est déjà une avancée considérable. Les expériences de YOUM 2 et YOUM 3 devraient grandement m'aider à trouver un tel algorithme s'il existe.

Mercredi 18 Avril

Et hop, un peu d'optimisation : Quelques problèmes corrigés et j'ai commencé à optimiser. On est encore loin des performances de YOUM 2, mais bon.

Mardi 17 Avril

Et hop la semaine va y passer : Ca corrige, ça corrige. J'espère pouvoir m'occuper des performances à partir de la fin de semaine si tout va bien. Mais je n'attends rien d'extraordinaire.

Les performances à la trappe : avec la première version de YOUM 3 les performances étaient en berne d'un facteur 1.2 environ. Avec cette nouvelle version on est plus sur une base de 1.9 ce qui est beaucoup.

Différence fondamentale entre YOUM 2 et 3 : YOUM 2 est sans compromis pour les performances. Cela a impliqué de ne pas gérer les cas trop tordus et exotiques souvent rares qui auraient nécessité beaucoup de puissance pour très peu de résultats. YOUM 3 c'est le contraire l'algorithme est conçu pour trouver toutes les associations de mots possibles. De plus il se paramètre avec une seule variable pour déterminer la longueur des syntagmes et le surcoût de puissance est très faible.

Des syntagmes trop longs n'ont pas de sens : De combien de mots doit être composé au maximum un syntagme ? Pour le meilleur rapport taille - performance ce sont des syntagmes de 4 mots. C'est le cas de YOUM 2. Avec YOUM 3 on est à 5 mots. Plus n'est pas nécessaire à mon sens car même s'il existe des syntagmes de 6 voire 7 mots, on peut sans risque d'erreur n'utiliser que 5 mots de ces syntagmes pour les référencer. Avec 4 ce n'est pas une certitude.

Lundi 16 Avril

Plus il y en a, plus il y en a ! : Pas mal de corrections faites, mais il en reste encore autant. Les jours suivants risquent d'être intéressants. En fait le débogage est la partie de la programmation que j'aime le plus.

Vendredi 13 Avril

Je suis le BTP (bâtiment travaux public) d'Onversity : J'ai adapté l'algorithme, maintenant il ne reste plus qu'à faire le ménage en ajoutant ce qui manque dans les règles et surtout en les corrigeant. Les règles sont les conditions qui déterminent quand il faut ajouter des mots à la queue de gestion des syntagmes, quand il faut la vider et comment. Que faire avec les balises HTML, les ponctuations, les apostrophes, les espaces blancs, ...

Mercredi 11 Avril

Une alpha pas si finale que ça : Je voulais voir pourquoi YOUM3 détectait parfois moins de mots que YOUM2 alors que cela ne devrait jamais se produire. Bien m'en a pris car je me suis rendu compte que l'algorithme utilisé n'est qu'un algorithme prototype visant à tester le principe de fonctionnement de cet algorithme. Non seulement il ne linke pas tous les mots vers le dictionnaire, mais il favorise les noms des définitions composées d'un seul mot et non de plusieurs. Ce dernier cas étant plus long et plus complexe à mettre en oeuvre.

Je voulais vous parler des différences entre YOUM 3 et 2, mais là j'ai clairement beaucoup de boulot encore sur la planche. Ce sera pour une autre fois.

Mardi 10 Avril

YOUM3 joue les speedy : Contrairement à YOUM2 qui reste entre 15% et 20% plus rapide, YOUM3 n'est pas bardé d'optimisations de code en tout genre. Le code est court et plus efficace que YOUM2, mais salement plus subtil des fois. J'ai pris plus de temps pour me rappeler son fonctionnement et faire quelques améliorations de code qui ne modifient pas la complexité du code.

Lundi 9 Avril

YOUM 3 sur le pont : J'ai repris le débogage de YOUM3. L'algorithme rentrait dans une boucle infinie sur certains textes. Il manquait une fonction que jamais l'algorithme ne trouvait :D. Pfff, pas simple comme algorithme, j'ai mis du temps à me remettre superficiellement dedans.

La raison pour laquelle je me replonge dans le développement de YOUM est que j'ai en préparation un article qui nécessite de faire fonctionner YOUM3.

Mercredi 4 Avril

Little Boxes : Le découpage a commencé et c'est la grosse rigolade. Le but est de n'utiliser que des boîtes (<div>) et nombre des tableaux. Les tableaux ne seront toutefois pas exclus car, pour les données tabulées comme celles de la liste des actualités, les tableaux seront toujours présents. Mais définir une organisation générale des différents blocs section de la page n'est pas inintéressant même si c'est parfois un peu casse tête. J'en profite évidemment pour mettre la nouvelle présentation qui, depuis le prototype, a bien changé.

Je vous donnerais le schéma des boîtes de la page d'accueil, ça vous donnera une idée. Enfin dès que je l'aurais fini.

Lundi 2 Avril

Quel accueil ! : Voilà, j'ai repris le prototype de la page d'accueil. Quoiqu'il ne va pas trop me servir vu que tout change ou presque. Ah si je garde l'organisation très modulaire que j'avais commencé. Je prends mon temps parce qu'à partir de cette page d'accueil bon nombre de fonctions utilisables dans les autres scripts d'affichage seront utilisées et comme je n'ai pas envi de les réécrire 15 fois (même si statistiquement je suis sûr de le faire pour une partie d'entre eux) j'essaye de penser aux détails. Pour l'instant j'en suis à des questions d'organisation de découpage de la page tant du point de vue graphique que xhtml.

Jeudi 29 mars

J'ai pris du retard avec les textes sur Onversity. Je laisse un peu de côté la prog. Sinon rien de spécial.

Mercredi 28 mars

Fin des scripts pour les articles avec les scripts Art_val et Art_deval. La prochaine étape c'est la réalisation d'une version alpha pour la page d'accueil. Cette page me permet de faire un premier test de fonctionnement des bases de données. Elle est donc très importante. C'est aussi la première page à avoir le décor, le vrai pas celui des scripts d'administration qui est bcp plus simple.

Mardi 27 mars

Réalisation des scripts Art_mod1, Art_mod2, Art_mod3, Art_mod4 de gestion de modification d'articles. Quatre scripts d'un coup, j'ai la pêche

Lundi 26 mars

Réalisation des scripts Art_sai1 et Art_sai2 de gestion de saisie d'articles.

Vendredi 23 mars

Ca avance. Si, si : J'ai terminé les scripts d'actualité aujourd'hui et j'ai fait le premier script des articles. Les articles sont, après les qcm, les scripts les plus compliqués. Effectivement le texte inclut non seulement le texte de chaque chapitre, mais aussi celui des titres de chaque chapitre qu'il faut savoir distinguer du texte, mais aussi leur organisation en chapitre, sous-chapitre, ... Il faut donc faire des procédures de lecture spéciales et aussi de contrôle pour s'assurer que tout est bien structuré par l'auteur avant d'enregistrer, et de lui permettre de le contrôler avant publication. En même temps ça doit rester le plus simple possible. Tant au niveau de la structure des enregistrements des articles que pour la saisie pour les auteurs.

Après les articles, je referais la page d'accueil pour la passer en mode alpha. Je ne dispose actuellement que d'un prototype de script pour la page d'accueil.

Il y a eu aussi quelques améliorations des scripts précédemment passés en beta, mais rien de bien important.en fait. Pour être plus précis, chaque jour pour l'instant j'ai des retouches à faire sur les scripts précédents.

Jeudi 22 mars

Journée faste : les scripts act_mod3, act_mod4 et act_sai1 sont passés beta.

Mercredi 22 mars

Le train train : Le script act_mod2 est passé beta. C'est un script écrivant dans des bases de données. J'en ai profité pour vérifier d'autres scripts d'écriture, car leur gestion d'erreur était insuffisante.

Mardi 20 mars

Ah, je vous jure : En faisant le script act_mod1, je me suis aperçu que les scripts aut_mod1 et pro_mod1 n'avaient pas leur code Xhtml à jour. Act_mod1 est passé beta.

Lundi 19 mars

Actualité nous voilà ! : Les scripts act_valid et act_deval sont passés en version beta. Rien de spécial à signaler.

Dimanche 18 mars

Del et New sont sur un bateau : Les scripts aut_del et aut_new ont été modernisés et passés en beta de même que les scripts pro_del et pro_new. Ceci met fin à une importante série de modification visant à moderniser la manière d'écrire le code des scripts d'Onversity. Cela inclut un nouveau code Xhtml, de nouvelles mesures de sécurité internes et externes, une gestion des erreurs plus efficace. Tous les scripts en statut alpha seront à modifier. Cela termine aussi le développement des scripts de la gestion d'administration des auteurs et des utilisateurs.

Les prochains sur la liste est la gestion des actualités qui sont en mode alpha actuellement et devront être entièrement mises à jour puis les articles qui faudra créer. La gestion des articles est très différente de la gestion actuellement en ligne. L'actuelle gère un enregistrement par paragraphe mais cela implique une grande complexité avec le moteur de recherche ou pour la gestion en ligne. Dans la nouvelle version, les articles seront un seul et même enregistrement de 80 ko au maximum. La gestion des interviews étant similaire à celle des articles, cette partie sera rapidement développée.

Lundi 11 mars

Ce cookie ne sent pas très bon : Je viens de m'apercevoir que les cookies de Luxor ne sont pas

sécurisés. Il faut qu'ils soient cryptés pour éviter qu'un malin ne puisse s'approprier facilement le compte d'une personne en créant soit même un cookie pirate. L'actuelle version d'Onversity dispose de fonction de cryptage, mais elle m'a vraiment créé trop de problème. Je dois la refaire en conservant la même page. Ensuite il faudra que je l'applique à tous les scripts qui allaient passer en version beta.

Dimanche 17 février

Enfin, ça va mieux. Après avoir attrapé deux maladies des bronches dont une coqueluche qui revient sur Strasbourg avec les pays de l'est je vais pouvoir me remettre au travail. En fait j'ai déjà commencé cette semaine.

Le et commercial (&) est un farceur : Pour être conforme au XHTML 1.0, il faut remplacer tous les caractères spéciaux par des codes spéciaux. Pour le '&' c'est '&'. Un code spécial commence par un '&', suivi du nom du code et se termine par un ';'. Onversity utilise déjà les codes spéciaux mais deux problèmes persistent : leur utilisateur dans les textes n'est pas généralisé dans toutes les situations et surtout ils ne sont pas utilisés dans les url. Je suis en train de changer cette état de fait, mais je me suis fait avoir avec les url. Effectivement ça ne concerne que les url intégrés dans le code xhtml, pas les url de redirection c'est-à-dire celles générés par le code C directement. Donc bingo je dois reprendre quelques dizaines de scripts. Pas méchant mais ennuyant.

La méthode Couet : Si j'ai les grandes lignes de ce à quoi doit ressembler en terme de qualité de développement et de fonctionnalité Luxor, pour certains détails ça reste souvent flou. Donc à chaque étape de nouvelles idées arrivent ou exceptionnellement disparaissent. Des améliorations de code surviennent soit en terme de performance ou de qualité. Le seul problème avec ça c'est que ça oblige à reprendre sempiternellement les mêmes bouts de code qu'à force on ne peut plus voir. Je ne travaille évidemment pas sur les 54 scripts déjà faits dans un premier jet, mais sur une dizaine. La bonne nouvelle c'est que ces scripts commencent à être assez léchés pour penser que je vais bientôt pouvoir enfin recommencer à avancer plus rapidement.

Les présidentielles : Ayant une activité politique avec les présidentielles celle-ci a augmenté. Je dispose donc de moins de temps jusqu'au 2 mai date du deuxième tour.

Dimanche 17 février

No time for ctime : Le gestionnaire d'erreur des scripts d'administration inclut à présent une fonction d'enregistrement de log. Cela fait défaut au gestionnaire d'erreur actuel d'Onversity. J'en ai profité pour résoudre un truc qui m'ennuyait. Pour avoir la date, je passe par les fonctions time et ctime. Or ctime ajoute un caractère de contrôle de fin de ligne. C'est bien si on met la date en fin de ligne pour le fichier de log, mais sinon c'est chiant. Donc suppression de ce caractère. Correction apportée aussi au gestionnaire de log d'erreur d'Hydrogen.

Jeudi 14 février

Trop d'erreurs tuent l'erreur. J'ai réalisé le script `admi-error` qui est le gestionnaire d'erreur des scripts d'administration. Ce script informe de l'erreur et propose de revenir à un niveau sans problème. Je dois lui ajouter une fonction d'enregistrement sur disque des erreurs comme je le fais avec `Hydrogen` de manière à voir les scripts avec lesquels les internautes ont des problèmes. Il faut savoir que la plupart du temps les internautes ne remontent pas les problèmes d'un site. En fait le but d'un gestionnaire de ce type est de permettre de trouver des bogues car oui il en reste toujours.

Mardi 13 février

Un script de plus. Le `pro_sai2` s'occupe de la validation de saisie d'un profil utilisateur.

Contrôle, contrôle. J'en ai profité pour modifier les contrôles de saisie. Au lieu d'essayer d'en faire un qui soit multitype, j'en fait un pour chaque type. Url, mot de passe, avatar, ... Dans `Luxor`, il y a deux types de contrôle. Les contrôles et les filtres. Les premiers s'occupent de déterminer si une chaîne est correcte avant d'être affectée à une base de données ou à un paramètre de script. Les seconds modifient l'entrée pour ne laisser sortir du filtre que ce qui est autorisé, plutôt utilisé, dans les forums.

Et hop, une optimisation. J'en ai profité pour faire une optimisation intéressante des fonctions de contrôles.

Lundi 12 février

Modification de la structure du fichier profil. Ajout du champ email et nombre d'actualités affichées simultanément. Il faut savoir que le fichier profil existe actuellement. Mais c'est juste une surcouche à 2 autres fichiers. Dans la nouvelle version il n'y aura plus que le fichier profil pour gérer les informations des utilisateurs.

J'ai commencé la création du script permettant l'enregistrement des données des profils utilisateurs. Le script de saisie étant déjà fait. Une fois ce script fait, le reste devrait être plus rapide pour les profils. A moins d'avoir encore oublié un truc. Ce qui à ce niveau de développement ne m'étonnerait pas :D.

Dimanche 11 février

Deux scripts de l'administration des profils utilisateurs ont été créés. J'ai commencé aussi à propager une modif de gestion du code Xhtml par le code C. La modif implique que le code xhtml généré par le code C puisse être transféré dans un fichier CSS de manière à gérer les skin par ce fichier. Précédemment, pour gérer la largeur de certaines classes CSS, j'utilisais un paramètre C, cela impliquait que chaque script n'avait pas nécessairement le même code CSS. Ce qui est incompatible avec une gestion de skin simple.

Samedi 10 février

Deux semaines de misère : Je n'étais pas satisfait des premières maquettes. Enfin première c'est une façon de dire la nième version d'une nième phase. J'ai donc repris cela et j'ai passé le temps qu'il fallait. 2 semaines. Le résultat est meilleur. Plus propre, plus accrocheur. Il faudra que je laisse décanter quelques semaines. Les forums me semblent vraiment bien partis. Je ne vois pas ce qu'il faudrait corriger. La page d'accueil par contre pose encore un petit problème d'organisation. Il faudra que je vois plus tard si je peux le régler comme je le souhaite.

Samedi 27 janvier

Buffer overrun : lorsque des informations proviennent de l'extérieur, il faut tester leur taille pour voir si elles sont compatibles avec la taille des variables qui doivent les recevoir. Dans le cas contraire, on s'expose à des plantages ou pire à des piratages. Je viens de l'inclure dans la gestion des cookies pour Onversity 2007. Les paramètres passés par les scripts disposent déjà d'un tel contrôle.

Aut_sai1 vient de passer en bêta. C'est le premier script. Youpi ! Il a en charge l'affichage du formulaire de saisie des auteurs. Comme les scripts des auteurs et des profils utilisateurs sont très semblables, Pro_sai1 va bientôt suivre. On devrait donc avoir rapidement une dizaine de scripts d'administration en bêta. Une fois cela fait, je pourrais terminer les scripts de sécurité et les passer en bêta aussi.

Vendredi 26 janvier

3 niveaux de sécurité sinon rien : L'accès au script d'administration se fait avec 3 niveaux de sécurité. 2 mots de passe disposés dans des bases différentes et un contrôle de droit d'accès écrit en dur. Cela implique que j'exploite la base de données des auteurs et des profils utilisateurs. Je viens de créer la base des profils. Elle est un peu plus étoffée que l'actuelle.

Jeudi 25 janvier

100 fois sur le métier remets ton ouvrage : J'avais validé les premiers scripts pour le Xhtml et le CSS, mais je me suis fait avoir. Je n'avais pas testé avec suffisamment de navigateurs. J'ai du modifier le code des scripts. Décidément l'acquisition d'expérience en Xhtml et CSS prend un peu de temps. Une fois que je serais rodé, ça ira mieux.

Mercredi 24 janvier

43 scripts défricheurs : Avant de commencer ce journal, j'avais créé 43 scripts d'administration pour un premier jet, histoire de voir comment j'allais articuler le code de la nouvelle version. Ces scripts m'ont aussi permis de créer un embryon de page d'accueil pour tester la nouvelle présentation. Mission réussie. Je suis donc passé à l'étape suivante : prise en compte du format

XHTML 1.0 et CSS et du 64 bits. Je reprends donc le code de ces scripts pour les rendre conforme au cahier des charges de Luxor.

Halte aux variables :J'ai commencé à remplacer les & par des & dans 2 scripts d'administration pour voir la réaction des navigateurs et des fonctions d'analyse. Mais en fait l'idée générale sera de limiter au maximum l'utilisation de variables dans les URL des scripts. Il y a un gros travail dans ce sens qui a été fait avec la version actuelle. Luxor poussera plus loin cela et surtout dans les forums.

Mardi 23 janvier

8 navigateurs ou rien : Pour tester la compatibilité des navigateurs, j'utilise Opera 9, IE7, Firefox 2 et Netscape 7.2. J'ai ajouté à la liste Avant Browser basé sur le moteur d'IE6 et Amaya le navigateur du W3C. Ce dernier est d'une lenteur et a du mal à faire fonctionner la majorité des pages web. Bref pas possible de naviguer avec, mais il intègre un analyseur du code Xhtml très utile. Ah il y a aussi ICEBrowser, un navigateur java. Rigolo, pas très rapide. Pour Safari, je passe par un émulateur disponible sur le net. Il est important de faire des tests primaires au moment du lancement de la production d'un site comme Onversity. Cela évite de propager des règles à des dizaines de scripts qui pourraient être erronés. A la fin du développement, on refera à nouveau les tests de compatibilité.

Ah le CSS ! Il y a vraiment des oublis dans la conception du CSS. Je veux bien remplacer toutes les balises <table> par des <div> mais encore faudrait-il que celles-ci soient faciles à centrer. Or ce n'est pas le cas car il n'existe pas de commande pour le faire au contraire des balises <table> avec align=center. Il existe bien une astuce que voici

```
#box_center {text-align:left; width: 800px; margin: 1 auto}
```

le problème est que si le contenant est bien aligné, pas le contenu qui lui est aligné a gauche. La solution consiste pour l'instant en

```
#box_center {text-align:left; width: 800px; margin: 1 auto;}  
#text_center {text-align:center;}
```

```
<div id"box_center">
```

```
<div id"text_center">
```

```
...
```

```
</div>
```

```
</div>
```

Samedi 20 janvier

- De l'organisation avant-tout : Un problème m'a amené à repenser l'organisation en module des scripts d'Onversity. Effectivement, si le programme principal compose le script, pour fonctionner, il a besoin de fonctions qui sont réparties dans des modules indépendants. Hydrogen, Youm,... sont des modules. Le problème c'est d'éviter de devoir faire appel à tous les modules pour la moindre ligne de code. Après diverses difficultés, j'ai décidé de créer un nouveau module : Xhtml1.c qui regroupera toutes les fonctions ayant une mission d'affichage de

balise xhtml ou de code CSS. Ça ne résout pas toutes mes difficultés, mais ça sera mieux ainsi vu que dans la version Luxor le code xhtml et CSS a une place plus importante que dans les versions précédentes.

Mardi 16 janvier

- Hydrogen 1.5 est passé en statut Alpha.
- Des problèmes de norme des entrées/sorties. Windows adhère par VC8 à la norme ISO C++ alors que GCC à POSIX. Résultat un close() est considéré comme obsolète et VC8 recommande _close. Bah, comme sous Windows ce n'est que pour les tests je laisserais en l'état. Si ça m'énerve trop je ferais des #if pour le précompilateur.
- Des warning inutiles. VC8 est beaucoup plus balèze sur la détection de problèmes potentiels qu'auparavant, mais parfois c'est lourd. Ainsi une copie d'une chaîne écrite en dure provoque le warning de buffer overflow.

Lundi 15 janvier

64 bits, je t'aime moi non plus.

Problème de déclaration : GCC et Microsoft n'ont pas la même position face au statut des déclarations int et long. Ainsi pour Microsoft int et long reste en 32 bits dans un environnement 64 bits, alors que pour GCC int reste 32 bits (ouf) mais long passe en 64 bits.

Stratégie de compatibilité : Le code d'Onversity étant 100% 32 bits et pouvant le rester sans problème (pas d'utilisation de base de données nécessitant 64 bits d'adressage, pas de calcul entier sur 64 bits, ...), j'ai opté pour n'utiliser que des déclarations int et unsigned int pour les entiers. Ce code sera portable pour GCC et VC8 pour environnement 32 bits et 64 bits normalement

Conséquence sur les performances

L'utilisation de variables 64 bits implique que les fonctions processeurs manipulant ces variables sont plus grandes. Cela peut générer plus de cache miss pour le cache L1 d'instructions. L'utilisation de variables 32 bits n'est donc pas un mal. La seule chose que je ne sais pas c'est si cela permet quand même de bénéficier des 8 registres supplémentaires. Normalement oui, mais je n'ai pas encore eu confirmation de cela.

Dimanche 14 janvier

Ouverture du journal pour la future version d'Onversity. Le nom du projet est Luxor en référence au temple de Luxor construit en Egypte au bord du Nil. A l'avenir les version auront des noms de haut lieu Egyptien.

