FROBNICATE

Special "The World Is Going To Hell!" issue.

Le meilleur magazine pour RISC OS!

We're baaaaack!!!

- Modifier votre code pour ARM32
- Premiers pas avec DeskLib
- Ovation

Hissing Spinach
Printemps 2017 Édition 31 €0

Contains sustained moderate geekery in French.

Indice:

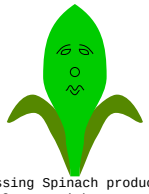
[#31, 2017/02/19 23:50 CET]

Remerciements:

Conçu et crée par Richard Murray.
La poésie en Español ecrit par Stephanie-Jane Murray.

Crée en association de **riscos.fr**, le première site web française (et gagnant de le prix pour Meilleur ressource en langue étranger *deux* fois! (2014 et 2015)).

http://www.riscos.fr/



A Hissing Spinach production
© 2017 Rick Murray

Crée *entièrement* sur un Raspberry Pi avec RISC OS et:
OvationPro, OpenVector, CrossStar, Snapper, ChangeFSI, et PrintPDF.

Dude, what's with the Froggie?

It's really rather simple, actually. Britain was not interested in hearing my voice during the farcical so-called "democratic referendum", just as the "democratic will of the people" seems to conveniently forget an entire *country* that voted in favour of Europe.
The British did not want to hear me then.
They do not get to hear me now.
This is my protest.

As such, if you cannot read French (especially my version of it!), go ahead and delete this file. There are no articles in English in the rest of this document.

I do not know what the future holds, but I know two things:
1. My life is here now, not in the UK.
and:
2. I am deeply shamed to be British. Every day, another news story that makes me cringe.

The English are choosing their destiny (and dragging the Northern Irish and Scottish along with them). Me? I'm choosing my destiny too. It's *not* England. Nor will it ever be.

I know that Frobnicate #30, in 2008, was to be the final issue. Almost another decade has passed. I found a job that I like in a company that seems to continually defy the gloomy economic forecasts. We're making more stuff than ever, and an ever increasing amount of our products are being sold internationally - we sell to sixty countries now, across the world. This is an international global economy. Everything interacts. Donald Trump is going to learn that, and so to are the British. You can't walk away and revive the ghost of the 1950s, that world no longer exists.
I was not planning on writing another issue of Frobnicate, however the irrepressible David Feugey of riscos.fr talked me into it. And with Brexit on the horizon, it seemed like a good opportunity to see how much my French sucks!

Rick, 2017/02/05

C'est bien facile, actuellement. L'angleterre n'était pas intéressé d'écoute ma voix dans le "référendum démocratique" stupide, comme le "décision démocratique de les gens" apparement oubliée un *pays entière* qui votée pour La Europe.
Les anglaise n'était pas intéressé de m'écoute a ce temps.
Ils n'arrivent pas de m'écoute maintenant.
Voici, mon protestation.

Donc, si on ne peut lire la française (notamment ma version!), aller supprimer cet fichier. Il n'y a pas des articles en anglaise dans le reste de cet document.

Je ne sais pas que suivre dans l'avenir, mais, je sais deux chose:
1. Mon vie, maintenant, est ici en France.
et:
2. Je suis vraiment honte d'être britannique. Chaque jour, un autre histoire dans les journaux que me fait reculer.
Les anglais choisis leur destin (et tirer les irlandais de nord et les écossais dans le même direction).
Moi? Je choisi mon destin aussi. Ce n'est *pas* l'angleterre. Il sont jamais l'angleterre.

Je sais que l'édition 30 de Frobnicate, en 2008, est le dernière édition. Presque un dizaine des années sont passé. Je trouve un travail que j'aime dans un entreprise qui toujours marche dans un direction contre de les prévisions économiques pires. Nous fait encore plus des produits aujourd'hui, et nous vendons nos produits à soixante pays partout le monde. C'est un économie internationale. Un interaction. Bientôt, Donald Trump et les anglais apprendra cet leçon. Il n'est pas possible de aller marche loin et ressusciter le fantôme de les années '50, c'est un monde que, simplement, n'existe pas.
J'ai pas d'intention de créer un autre édition de Frobnicate, mais le génial David Feugey de riscos.fr me demande. Et, avec Brexit très bientôt, je croire qu'il est un superbe opportunité de voir comment mauvais est mon française. *Bon courage!*

Rick, 2017/02/05

Modifier votre code pour ARM32

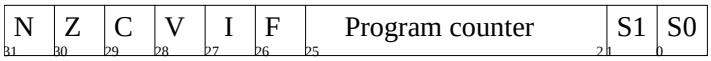
Dans cet article, nous pensons comment modifier votre code dans le langue d'assemblage pour le nouveaux génération des processeurs ARM en 32 bits.

Avant, un peu d'histoire:

Il était un fois... le première processeur ARM arrive en 1983, et le première ordinateur avec un ARM arrive en 1987. Le taille des registres et l'accès mémoire sont toujours 32 bits (quatre octets), mais les processeurs anciens sont souvent appelé "26 bit". Pourquoi?

Le solution est de voir le 15ème registre, c'est l'un qui contient l'adresse de l'instruction exécutée par le processeur. Dans les années '80s, les quantités énormes de mémoire vive n'existe pas. Le BBC Micro contient 32Ko. Un PC typique contient autour de 640Ko~1Mo. Le première machine ARM, l'Archimedes, contient entre 512Ko (A305) vers 4Mo (A440).
Pour cet raison, il est anticipe que il suffit pour l'ARM d'être capable d'adresser 64Mo de mémoire. Souviens que le controlleur de mémoire (MEMC1a) parler à que 4Mo. Pour plus que 4Mo on a besoin des plusieurs MEMCs (une vrai acte de précarité!).

Donc, registre 15 est comme:



S0 et S1 décrire le "mode" de le processeur (USR, FIQ, IRQ, ou SVC).
Le reste, NZCVIF compris le registre d'état.
I et F est les drapeaux pour l'inhibition des interrompre (normale ou vite).
N est pour un résultat négatif. Z est pour un résultat de zéro (si 4 est égal de 4, le résultat d'un comparaison est nul, ou zéro (aucun différence)). C est pour un retenue (Carry en anglais). Enfin, V est pour un dépassement de capacité (oVerflow en anglais).
Bits 0 et 1 sont toujours zéro à cause de le processeur marche avec les instructions de 4 octets. Six drapeaux existe en les bits 26 vers 31. Ça qui reste? Les 26 bits pour l'adresse de l'instruction.

C'est bon. C'est puissante – un seul instruction est capable de reviens d'un fonction et restaurer tout l'état de le processeur, comme:

```
LDMFD  R13!, {R0-R12, PC}^
```

Mais, alors, sur mon téléphone portable, j'ai un app qui me donne les prévisions météo qu'est plus gros que 64Mo. Aïe!
Comment c'est possible?

Facile. En ARM32 (contre ARM26), le registre 15 est uniquement pour l'adresse. Avec 32 bits, il peut adresser jusqu'au 4Go.
Et l'état? Il bouge vers un registre spécifique – le CPSR (Current Processor Status Register, ou registre d'état de processeur courant). Le 'courant' est parce que les modes privilégié possède deux PSRs, le *courant* et aussi le *sauvegardé*.

Donc, le plus gros travail pour adapter votre logiciel pour ARM32 est de supprimer tout les choses avec le PSR dans registre 15.
Actuellement, il est un peu plus facile parce que le nouvelle API (protocole d'applications) pour RISC OS sur ARM32 dictée que il n'est pas de besoin de restaurer les drapeaux. Donc pour le plupart il suffit de simplement fait exactement ça:

Si on voit:

```
MOVS   PC, R14
```

enlève le 'S' pour ne restaurer pas les drapeaux:

```
MOV    PC, R14
```

Même pour retrouver les registres de le pile:

```
LDMFD  R13, {Rx-Ry, PC}^
```

enlève le '^' pour ne restaurer pas les drapeaux:

```
LDMFD  R13, {Rx-Ry, PC}
```

Voilà. Pour les applications simples et des modules simples, ça compte pour le plupart des problèmes.
Le reste? Tourner le page...

Parce que le PSR n'est pas avec PC en registre 15, il est logique que le suivant ne marche plus:

```
ORRS    PC, LR, #(1 << 28)
```

Cet instruction copié LR (l'adresse de retour) dans PC (l'adresse actuel) et restaurer les drapeaux *et* aussi mettre le drapeau 'V' actif. C'est normale pour signaler un erreur.

Sur ARM32, on utilise les instructions MRS et MSR pour copier le CPSR vers un registre, et un registre vers le CPSR.
Techniquement, le code suivent fait le même chose (et corrompu R1):

```
MRS     R1, CPSR
ORR     R1, R1, #(1 << 28)
MSR     CPSR_flg, R1
```

Ou, si les autres drapeaux ne sont pas importantes, on peut utiliser que:

```
MSR     CPSR_flg, #(1 << 28)
```

Mais, *attende*! C'est bien pour les processeurs avec MRS et MSR. Mais pour les plus anciens?

Franchement, je pense le meilleur réponse est ***tant pis*** parce que les machines plus anciens sont comme un quart de siècle de ancienneté. Il est comme anticiper pourquoi le logiciel de aujourd'hui ne marche pas avec Windows 3.11!

Mais, il existe un moyen facile pour mettre le drapeaux V sans corrompu registres, qui marche sur tout les processeurs avec RISC OS:

```
CMP     R0, #(1 << 31)
CMN     R0, #(1 << 31)
```

Pour le code dans un module, peut être il changer le mode de le processeur comme suivant:

```
MOV     Rx, PC
ORR     Ry, Rx, #%11    ; SVC
TEQP    Ry, #0
```

et pour restaurer le mode originale:

```
TEQP    Rx, #0
```

Voici comment fait le même chose sur ARM32:

```
MRS     Rx, CPSR
BIC     Ry, Rx, #&1F
ORR     Ry, Ry, #%10011 ; SVC32
MSR     CPSR_all, Ry
```

Le BIC effacé le bits pour le mode, puis le ORR ajouté le nouveau mode.
Pour restaurer l'ancien, c'est que:

```
MSR     CPSR_all, Rx
```

Il n'existe pas de moyens de fait cet type de chose sur tout les processeurs. Mais, c'est pas tout perdu parce que il est pas difficile de détermine si le processeur marche comme ARM26 ou ARM32. Rappel que dans un comparaison en ARM26, le première registre jamais compris les drapeaux, mais le deuxième registre compris les drapeaux.
Donc:

```
TEQ     R0, R0    ; mettre Z actif
TEQ     PC, PC    ; mettre Z actif si ARM32
```

Un explication. Le dernière TEQ fait le comparaison suivent:

ARM26: PC vs PC+PSR
ARM32: PC vs PC

Donc, le résultat est EQ pour ARM32 et NE pour ARM26.

Et le première TEQ? C'est pour assurer que un drapeau est actif avant. Parce que en mode USR avec aucun drapeau actif, le PSR est tout zéro. Si on mettre un drapeau actif avant, PC ne jamais arrive pareil que PC+PSR!

En effet, on peut utiliser tout de les idées pour créer code qui marche sur tout:

```
TEQ     R0, R0
TEQ     PC, PC
BNE     code_pour_arm26

; ici, code pour ARM32 avec
; MSR etc

B       continuation

.code_pour_arm26
; ici, code pour ARM26 avec
; TEQP etc

.continuation
; et après, tout viens ici...
```

Désolé, c'est pas le fin de notre histoire...

Choses à éviter:

- `MOVS PC`
- `TEQP`
- `LDM ^`
- `TST xx, PC`
- `ORR`, `ADD`, `AND` sur drapeaux en R14/R15

Chose à regarder bien:

- Le nouveau API dit que il n'est aucun SWI qui restaurer les drapeaux. Donc, le code suivant ne marche plus:

```
CMP     Rx, Ry
SWI     xxx
BEQ     autre_place
```

  Heureusement, code comme ca est vraiment rare.

Notre histoire n'est pas fini!

Les anciens processeurs ARM (jusqu'au ARMv5) fait un lire de mémoire avec un adresse *non aligné* (pas un mot, adresse divisé par quatre) dans un manière bizarre.
Bref, il *toujours* lire depuis un adresse *aligné*, mais si l'adresse de origine n'est pas aligné, l'ordre des octets dans le mot change leur position.

Pour exemple:

```
DIM code% 31
P% = code%
[ OPT 2
  LDR    R0, [R0]
  MOV    PC, R14
.word
  EQUD   &11223344
  EQUD   &55667788
]
A% = word   : PRINT ~USR(code%)
A% = word+1 : PRINT ~USR(code%)
A% = word+2 : PRINT ~USR(code%)
A% = word+3 : PRINT ~USR(code%)
```

Si on exécuté cet programme (sur un processeur qui fait "rotated loads"), le résultats sont... intéressant:

```
11223344
44112233
33441122
22334411
```

Sur les processeurs plus récent, le processeur fait exactement comme demandé:

```
11223344
88112233
77881122
66778811
```

Pourquoi 88 puis 11? Le processeur est normalement *little endian*, donc l'octet le moins important est le première en memoire. `word?0` est &44, pas &11.
Donc, littéralement, en mémoire, il arrive comme:

```
44 33 22 11 88 77 66 55
```

Voilà, "adresse plus un" donnera `33 22 11 88`.

Le fonction normale pour RISC OS est de créer un *exception* si le processeur rencontre un adresse non aligné. Dans cet cas, le programme arrêt avec le message *Internal error: abort on data transfer at &xxxxxxxx*
Ca dire que il y un exception pendant le transfert des données. Si cet erreur arrive, regarder si l'adresse n'est pas aligné.

Quand un erreur arrive, le système fait un copie de tout l'état de le processeur. On peut lire avec le commande `*ShowRegs` comme:

```
>*ShowRegs
Register dump (stored at &20009E50) is:
R0  = 0000904D R1  = 00000008 R2  = 656B696C R3  = 65687420
R4  = 4D524120 R5  = 2F332F32 R6  = 2F303136 R7  = 2C303137
R8  = 00008700 R9  = 00407FC8 R10 = 00000000 R11 = 00008100
R12 = 00008FE1 R13 = 00407FB0 R14 = FC1B07D0 R15 = 0000904C
Mode USR32 flags set: nzCvqjggggeAift       PSR = 20000110
```

Et le code? Soustraire 8 de le valeur de PC (pour le *pipeline*) et examiné l'instruction:

```
>*MemoryI 9044 +8
00009044 : ..'å : E5900000 : LDR     R0,[R0,#0]
00009048 : .ð á : E1A0F00E : MOV     PC,R14
```

Et bien, nous lisez le mot a l'adresse de R0 et place cet mot en R0. Donc R0 contient l'adresse. L'adresse? &904D. C'est bonne?

```
>PRINT &904D / 4
   9235.25
```

Non. C'est pas bonne. C'est un adresse non aligné.
Comment réparer ce chose? Ce dépende sur le logiciel. Désolé, c'est pas toujours facile...

Avec cet aide, nous peut modifier notre ancien logiciels pour le génération courante des ARMs!

¶

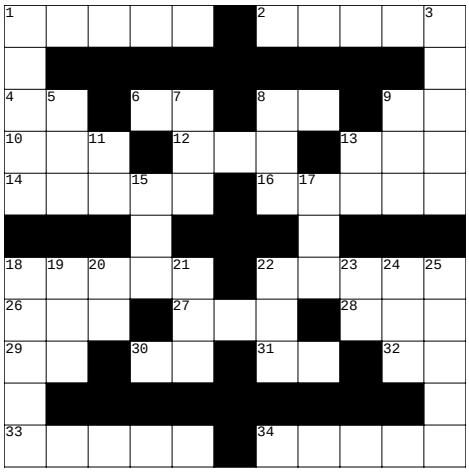
Promesas

Preguntas y respuestas
entre tú y yo
entre tante gente
que iban luego
después, solo, sin tí, amado
te digo en mis sueños
palabras y promesas
de cualquier y cuando
siempre tus deseos
siempre te digo cosas
entre tú y yo.

Stephanie-Jane Murray, 1990

Cassez votre tête!

Facile? C'est décevant! En plus, une description en italique signifie que le bon mot est en anglais! (^_^)



À travers

1. *Ours noir et blanc*
2. *Polices*
4. *Aller*
6. La, les, et ...?
8. *Vers le haut*
9. *Acronyme pour entrée/sortie*
10. Ce qui couvre 71% du monde
12. Trois points, trois traits, trois points
13. *Acronyme pour unité de soins intensifs*
14. *Jupe*
16. Chanteuse, ... Lavigne.
18. Brassica napus L. (sous-espèce napus)
22. *Après*
26. *Acronyme pour Acide ribonucléique*
27. "Big Blue" dans le monde des ordinateurs
28. Marque populaire de cuisinières en fonte
29. Hexadécimal pour '®' (en ISO 8859/1)
30. Unité de mesure de liquide
31. Le ça, moi, et surmoi: Le ça *en Latine*
32. *Acronyme pour surdosage*
33. *Miam, miam, miam. miam!*
34. Le muscle le plus important du corps

Vers le bas

1. *Feuilles d'un livre*
3. Ville principale de Corée
5. *Quercus*
7. Direction en face de Ouest
8. *Acronym pour États-Unis*
9. Pas là
11. *Acronyme pour interface utilisateur*
13. *Acronyme pour infrarouge*
15. Oryza sativa ou Oryza glaberrima
17. Animé, vivant, clair, brillant, etc
18. *Fou*
19. *1*
20. *Initiales: le 2ème ville des États-Unis*
21. Contre les vampires
22. "Friend" en française
23. Un adjectif possessif
24. La répresentation de la conscience de soi-même
25. Détection par ondes électromagnétiques

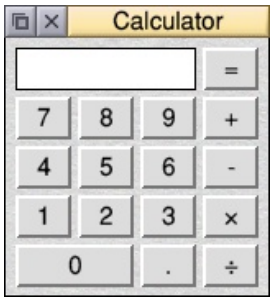
À gauche, le solution. Mais, attention, il est codé en ROT13! C'est un version du *chiffre de César*, où chaque caractère est décalé par treize positions, comme:

Originale: `ABCDEFGHIJKLMNOPQRSTUVWXYZ`
ROT13: `NOPQRSTUVWXYZABCDEFGHIJKLM`

Bon chance!

Premiers pas avec DeskLib

Dans cette article, nous découvrons comment écrire un logiciel avec DeskLib.



Conditions Préalables

Il y trois conditions préalables:

1. Le ***DDE*** de RISC OS. Le DDE est un produit commerciale, c'est pas gratuite, mais c'est le seul moyens de créer votre version de RISC OS customisé (c'est pas difficile, juste un peu lent).
Il existe un version de DeskLib (pas pareil) pour GCC, mais je jamais utiliser GCC.

2. ***DeskLib*** – vous pouvez télécharger mon version à: **`http://www.heyrick.co.uk/desklib/`** (cliquez sur le lien "*an unofficial release available if you'd like to play*" pour aller au zip file sans lire tout le blabla).

3. Vous comprendre le langue C. Voici, le parte le plus problématique! Si vous comprendre que BASIC... uh... c'est un joue vraiment différent.

Pour notre exemplaire, nous créer un calculatrice. Facile, mais il démontre comment utiliser DeskLib.

Créer les ressources pour l'application

Créer un nouveau dossier "!Calc". Dans cet dossier, créer aussi les dossiers "c" et "o", comme normale pour un logiciel en C.

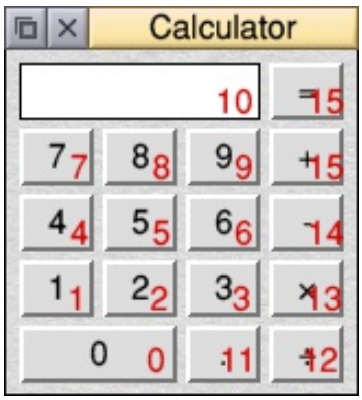
Créer le fichier "!Run" comme suivant:

```
| Run DeskLib example - Calculator
|
| 14th February 2017

Set Calc$Dir <Obey$Dir>
WimpSlot -min 64K -max 64K
Run <Calc$Dir>.!RunImage
```

Pour le magie, le *MakeFile*, et attention – le symbole ↵ signifie un ligne continue (pas deux lignes, comme apparait ici):

```
# Project:   Calculator

# Toolflags:
CCflags    = -c -depend !Depend -I,C: ↵
-throwback -fahn -apcs 3/32bit -Wcdp
ObjAsmflags= -APCS 3/32bit -desktop -Depend ↵
!Depend -ThrowBack -Stamp -IC:,
Linkflags  = -aif -o $@

# Source files
c_files    = o.main
s_files    =
libraries  = DeskLib:o.DeskLib32 C:o.stubs

# How to build
@.!RunImage: $(c_files) $(s_files) $(libraries)
             link $(linkflags) $(c_files)
$(s_files) $(libraries)

# A macro for building the C code
.c.o:;       cc $(ccflags) $< -o $@

# A macro for building the assembler code
.s.o:;       objasm $(objasmflags) -from $< -to
$@


# Dynamic dependencies:
```

Et, enfin, désigner votre Template avec des icônes numéroté comme indiqué en rouge.
L'icône blanc pour le résultat n'est pas une icône pour l'écriture. C'est que un étiquette.
Les autres? Boutons simples.



Le code source de l'application

Dans le fichier *!Calc.c.main*...

Jamais oublie de démarrer votre code source avec un texte qui décrire où, quoi, quand, etc. Si il y des dépendances (autres logiciels, etc) parle de les ici.
Attention! Le haut de votre code source n'existe pas uniquement pour le blabla légale. Oui, mettre ici les choses légales, mais toujours *toujours* aussi explique le raison pour cet fichier.
Notre petit projet contient un seul fichier. Les logiciels plus complexe peut contient dizaines des fichiers. Sans explication, c'est un cauchemar véritable.

```
/* Calculator v0.01

   Name   : Calculator
   Version: v0.01
   Date   : Tuesday, 14th February 2017

   Purpose: Demonstrate using DeskLib!

   Created by Rick Murray for Frobnicate issue #31

   Source code released under the EUPL.
*/
```

Suivant, nous a besoin d'ajouter tous les bibliothèques pour notre calculatrice. Comme CLib, les choses en DeskLib sont organisé par thématique. C'est pourquoi il y plusieurs (actuellement, c'est rien!).

```
// CLib
#include <ctype.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "kernel.h"

// DeskLib
#include "DeskLib:Core.h"
#include "DeskLib:Error.h"
#include "DeskLib:Event.h"
#include "DeskLib:EventMsg.h"
#include "DeskLib:Handler.h"
#include "DeskLib:Icon.h"
#include "DeskLib:Menu.h"
#include "DeskLib:Resource.h"
#include "DeskLib:Screen.h"
#include "DeskLib:Sound.h"
#include "DeskLib:Template.h"
#include "DeskLib:Time.h"
#include "DeskLib:Window.h"
```

Nos définitions de mémoire pour stockage. Le quatre en premier est pour DeskLib, le dernier cinq est pour notre application.

```
// Variables
static icon_handle iconbaricon;
static BOOL finished = FALSE;
static window_handle calculator_window;
static menu_ptr main_menu = NULL;

static double thisnum = 0;      // Number currently visible on-screen
static double thatnum = 0;      // Previous number that was on-screen
static double result = 0;       // Result of calculation
static char  display[16] = ""; // What's visible on the screen
static int   lastop = 0;        // Last maths operation
```

Noms prédefinis pour associer avec le mémoire "lastop" parce-que "_ADD" est plus compréhensible que "1".
Les *chiffres magiques* sont mauvais. À éviter!

```
// For lastop
#define _XXX 0
#define _ADD 1
#define _SUB 2
#define _MUL 3
#define _DIV 4
```

Décrire tous nos fonctions avant, c'est mieux comme ça (sinon, on peut utilise un fonction uniquement *après* le compilateur sais le fonction).

```
// Function prototypes
static void initialise_wimp(void);
static BOOL iconbar_handler(event_pollblock *, void *);
static BOOL menu_choice(event_pollblock *, void *);
static BOOL calculator_clickhandler(event_pollblock *, void
*);
static void append_number(int);
static void append_period(void);
static void do_add(void);
static void do_subtract(void);
static void do_multiply(void);
static void do_divide(void);
static void do_equals(void);
static void maths_op(void);
```

Enfin, notre point d'entrée dans l'application.
Nous appelons un fonction pour initialiser tout, puis nous mettre un zéro dans "l'écran" de notre calculatrice, et après le cycle pour le système de multitâche. Nous utilisons "Wimp_PollIdle" pour demander le système de nous appel uniquement quand quelque chose passe (et pas chaque cycle). Nous spécifie un temps de retour de 5000 centisecondes depuis "maintenant" (50 secs), mais parce que il n'est aucun gestionnaire pour l'événement "nulle", le système jamais nous appel sans raison.
Le cycle fini si "*finished*" est pas zéro (vrai), et dans cet cas, un peu de nettoyage avant que terminer.

```
// Program entry point
int main(int argc, char *argv[])
{
   argc = argc, argv = argv;

   // Start us up!
   initialise_wimp();

   // If display is "", we ought to show "0" in the display
   Icon_SetText(calculator_window, 10, "0");

   // Enter idling polling loop
   while( !finished )
   {
      Wimp_PollIdle(event_mask, &event_lastevent,
                    (Time_Monotonic() + 5000));
      Event_Process(&event_lastevent);
   }

   // We come here when finished, so tidy up and quit
   Template_ClearAll();
   Event_CloseDown();
   exit(EXIT_SUCCESS);
}
```

Le fonction pour démarrer tout...

```
static void initialise_wimp(void)
{
   int  wimpmsgs[] = { message_MODECHANGE, 0 };
```

Nous à besoin de répondre au message pour changement de MODE, mais pas d'autres.

```
   // Get stuff started and/or loaded
   Resource_Initialise("Calc");
   Event_Initialise3("Calculator", 310, wimpmsgs);
```

Les ressources existe sur "Calc", qui deviendra le symbole <Calc$Dir> pour trouver les templates, sprites, et cetera.
Puis, on créer l'appli sous le nom de "Calculator", ou si vous préfère, "Calculatrice"!

Après, les fonctions pour charger les images, démarrer le système des messages pour événements, enregistrer le MODE courant, initialiser les templates, indiquer de utiliser les images en "*resource_sprites*" pour les templates, et enfin ouvrir le fichier templates qui s'appelle "Templates":

```
   Resource_LoadSprites();
   EventMsg_Initialise();
   Screen_CacheModeInfo();
   Template_Initialise();
   Template_UseSpriteArea(resource_sprites);
   Template_LoadFile("Templates");
```

Tout les choses au dessus sont standardisé, mais on peut changer si il y besoin. Par exemple, on peut charger un fichier avec des images séparément pour les templates, et dans temps ancien (mois souvent aujourd'hui) on posséder les templates en façon "2D" et "3D".

Le prochain chose est de retrouver le fenêtre "calculator" de les templates, créer cet fenêtre, puis ajouter un gestionnaire pour si l'utilisateur clique sur notre fenêtre.

```
   // Create the window and attach click handler
   calculator_window = Window_Create("calculator", 45);
   if (calculator_window == 0)
      Error_ReportFatal(0, "Unable to load templates.");
   Event_Claim(event_CLICK, calculator_window, event_ANY,
               calculator_clickhandler, NULL);
```

Regarde bien. Il n y pas des troubles de répondre aux événements Wimp, chercher notre fenêtre, etc. Le système de gestionnaire des événements sur DeskLib fait tout de le travail pour nous.
Avec cet seul ligne de code, nous indiquer de quand l'utilisateur clique sur une icône dans notre fenêtre, donner les infos au fonction *calculator_clickhandler*.

Imagine, donc, le puissance de mettre plus des événements (clavier...) avec un fonction différent pour chaque. Voilà, c'est puissant et facile!
Mettre aussi des gestionnaires génériques pour les

événements de ouvrir, refait, et fermer les fenêtres. Comme toujours, nous peut créer nos fonctions pour raisons spécifiques, mais dans cet cas, les génériques suffit.

```
// Set up default handlers for Wimp events
Event_Claim(event_CLOSE, event_ANY, event_ANY,
            Handler_CloseWindow, NULL);
Event_Claim(event_REDRAW, event_ANY, event_ANY,
            Handler_NullRedraw, NULL);
Event_Claim(event_OPEN, event_ANY, event_ANY,
            Handler_OpenWindow, NULL);
```

Maintenant, créer le menu. Un seule chose, “Quit” (ou “Quitter” en française). Et, comme pour le fenêtre, rallonge un fonction pour l’événement de l’utilisateur clique un option sur cet menu.

```
// Create the main menu ("Quit", there is no "Info ->")
main_menu = Menu_New("Calc", "Quit");
Event_Claim(event_MENU,  event_ANY, event_ANY,
            menu_choice, NULL);
```

Parce que je suis nul à dessin, le ne créer pas un icône pour notre appli. Donc, quand nous démarrer, mettre un copie de l’icône de *!SciCalc* sur le barre des icônes. Et, toujours, un fonction pour cet événement.

```
// Stick an icon on the iconbar.
// WE STEAL THE !SCICALC ICON. :-)
iconbaricon = Icon_BarIconUser("!SciCalc",
   iconbar_RIGHT, (unsigned int *)resource_sprites);
Event_Claim(event_CLICK, -2, iconbaricon,
            iconbar_handler, NULL);
```

Dernière chose, un fonction pour l’événement de un changement de MODE.

```
// We need to respond to  mode change events
EventMsg_Claim(message_MODECHANGE, event_ANY,
               Handler_ModeChange, NULL);

return;
}
```

C’est important de saviez bien le différence entre programmation en BASIC et les grandes structures de CASE...ENDCASE, et l’idée de attacher un fonction au chaque événement et laisser le partie difficile au bibliothèque.

Voici, le première fonction pour un événement. Dans cet case, c’est pour un clique sur notre icône dans le barre des icônes. Quand nous entrer le fonction, nous perçu tout les données de “*event*” (tout qui est donné par le Wimp). C’est un peu plus compliqué que juste le bouton, mais il permet nous de accéder tous les infos disponible comme nous souhaite.
Pour le bouton, c’est *event->data.mouse.button* mais nous peut passe pour plus des détails et chercher pour *event->data.mouse.button.data.select* pour le bouton “Select”. Même pour *menu* et *adjust*.

Pour *Select* (bouton gauche), nous ouvrons notre fenêtre. Pour *Menu* (bouton centre), nous ouvrons notre menu (le -1 pour coordonne “y” pour signaler de mettre en bonne placement pour le barre des icônes). Et, enfin pour le bouton droite? Rien à fait.

```
static BOOL iconbar_handler(event_pollblock *event, void
*reference)
{
   reference = reference;

   // SELECT clicked?
   if (event->data.mouse.button.data.select)
      Window_Show(calculator_window, open_CENTERED);

   // MENU clicked?
   if (event->data.mouse.button.data.menu)
      Menu_Show(main_menu, event->data.mouse.pos.x,-1);

   return TRUE;
}
```

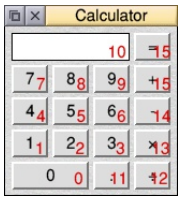
Le fonction pour le menu. Le bibliothèque remplir le valeur de *menu_currentopen* automatiquement avec le référence du menu visible sur l’écran. Donc, en peut utiliser cet valeur pour détermine un menu contre un autre. C’est plus puissant quand il y plusieurs menus. Et, le première référence de objet choisi est dans *menuitem*, retrouvé sur le données emis par le Wimp dans *event*.
Avec ces infos, c’est simplement une teste pour “quel menu” et “quel option”?

```
BOOL menu_choice(event_pollblock *event, void *reference)
{
   mouse_block ptr;
   int menuitem = event->data.selection[0];

   reference = reference;
```

```
   if (menu_currentopen == main_menu)
   {
      // Main menu
      if (menuitem == 0)
      {
         // Quit!
         finished = TRUE;
      }
   }

   // Keep menu open if Adjust clicked
   Wimp_GetPointerInfo(&ptr);
   if (ptr.button.data.adjust)
      Menu_ShowLast();

   return TRUE;
}
```

Il n’y pas de besoin pour garder le menu ouvert si on clique avec le bouton droite parce que le seul option est pour quitter l’appi, mais je vous donnera le code car il est bon de savoir. C’est le petite partie en fin là. Trouver l’état du flèche (et quel bouton) et si il est du bouton Adjust (droite), ouvrir encore le dernière menu (mêmes positions).

Le prochaine fonction est pour cliques sur notre fenêtre. C’est plus longue parce que il y seize icônes, mais le principe reste toujours le même chose. Quand nous arrivons sur notre fonction, nous sais que le utilisateur clique sur notre fenêtre, donc c’est que un besoin de examine quelle icône, et fait les necessaires.



Rappel, les icônes sont numéroté comme dans cet petit exemplaire.

```
static BOOL calculator_clickhandler(event_pollblock *event,
void *reference)
{
   int icon = event->data.mouse.icon;

   reference = reference;

   if ((event->data.mouse.button.data.select) ||
       (event->data.mouse.button.data.adjust))
   {
      switch(icon)
      {
         case  0 : // fall through
         case  1 : // fall through
         case  2 : // fall through
         case  3 : // fall through
         case  4 : // fall through
         case  5 : // fall through
         case  6 : // fall through
         case  7 : // fall through
         case  8 : // fall through
         case  9 : append_number(icon);
                   break;

         case 10 : if (event->data.mouse.button.data.adjust)
                   {
                      // Adjust-click display to blank it
                      display[0] = '\0';
                      Icon_SetText(calculator_window,
                                   10, "0");
                   }
                   break;

         case 11 : append_period();
                   break;

         case 12 : do_divide();
                   break;

         case 13 : do_multiply();
                   break;

         case 14 : do_subtract();
                   break;

         case 15 : do_add();
                   break;

         case 16 : do_equals();
                   break;
      }
   }

   return TRUE;
}
```

Le seul complication là, l’action sur l’icône des résultats passe uniquement si il est cliqué avec le bouton droite, Adjust.

C’est le fin de les gestionnaires d’événements. Maintenant, deux fonctions pour mettre les chiffres ou pointe décimale sur l’icône des résultats.

```
static void append_number(int num)
{
   // Append <num> to text in display icon
```

```
   int  end = strlen(display);

   if (end > 12)
   {
      Sound_SysBeep();
      return;
   }

   display[end] = num + 48; // number to ASCII
   display[end+1] = '\0';

   Icon_SetText(calculator_window, 10, display);

   return;
}
```

Une faute ici, c'est possible d'entrer "0000" comme un chiffre.

Pour un idée de reglé cet situation, on peut réfléchir sur le fonction pour le pointe décimale. Il bloque "...", mais – attention – le chiffre "0.00001" est valable!

```
static void append_period(void)
{
   // Append '.' to text in display icon, but only ONCE!

   int  end = strlen(display);
   char *per = 0;

   if (end > 12)
   {
      Sound_SysBeep();
      return;
   }

   per = strchr(display, '.');
   if (per != 0)
   {
      Sound_SysBeep();
      return;
   }

   display[end++] = '.';
   display[end] = '\0';

   Icon_SetText(calculator_window, 10, display);

   return;
}
```

Maintenant, les fonctions mathématiques. Si on penser d'une calculatrice traditionnelle, il accepte les commandes, mais il fait les calcules dans le commande suivant. Par exemple, si vous tappez 2 + 3 + 4 il reste "5" sur l'écran. Vous peut tappez sur "=" pour compléter le calcule et arrive au chiffre "9".

Notre application marche dans le même façon. Donc nous démarrer chaque fonction avec un appel de *maths_op()* (pour fait les calcules), puis nous mettre le nouveau type de calcule sur *lastop*.

```
static void do_add(void)
{
   // Got something to do?
   maths_op();

   // Set lastop to be an add
   lastop = _ADD;

   return;
}

static void do_subtract(void)
{
   maths_op();
   lastop = _SUB;

   return;
}

static void do_multiply(void)
{
   maths_op();
   lastop = _MUL;

   return;
}

static void do_divide(void)
{
   maths_op();
   lastop = _DIV;

   return;
}

static void do_equals(void)
{
   maths_op();
   lastop = _XXX;

   return;
}
```

Et le dernière chose? Le fonction *maths_op*.

Quand nous entrer, nous copie *thisnum* (le chiffre sur l'écran) vers *thatnum* parce que c'est le chiffre d'avant.
Après, nous refait *thisnum* avec le chiffre sur l'écran a ce moment.
Puis, nous regarder sur *lastop* pour détermine quel opération mathématique de fait. Le résultat arrive vers *result*.
Si l'opération est "égal", nous copier *thisnum* directement vers *result*.
Après, les contenants de l'écran sont annulés pour le prochain fois, et le *result* est visible sur l'écran. C'est pareil d'un calculatrice quand le résultat apparu sur l'écran, mais si on tappez un chiffre, tout sont remplacée pour le chiffre.

Et, enfin, copier *result* vers *thisnum* pour le prochaine calcul (*result* est uniquement temporaire).

```
static void maths_op(void)
{
   char disp[16] = "";

   // Shift <thisnum> into <thatnum>
   thatnum = thisnum;

   // Read <display> into <thisnum>
   thisnum = atof(display);

   // Do we have an operation to perform?
   switch (lastop)
   {
      case _ADD : result = thatnum + thisnum;
                  break;

      case _SUB : result = thatnum - thisnum;
                  break;

      case _MUL : result = thatnum * thisnum;
                  break;

      case _DIV : result = thatnum / thisnum;
                  break;

      case _XXX : result = thisnum; // so below code works!
                  break;
   }

   // Set <display> to empty
   display[0] = '\0';

   // Show the result on the screen
   snprintf(disp, 12, "%.3f", result);
   Icon_SetText(calculator_window, 10, disp);

   // Now put <result> into <thisnum> for next time
   thisnum = result;

   return;
}
```

C'est tout pour créer un calculatrice facile pour donnera un démonstration de DeskLib. Il est un bibliothèque un peu ancien (à ce moment, pas trop des fonctions pour les choses dans RISC OS 5), mais il reste toujours un bibliothèque sympa qui aide de fait votre logiciels.

Avec notre joli calculatrice, il existe des problèmes. Ils reste comme un exercice pour vous a régler:

• Pas de sprite pour l'application.

• Besoin d'un icône pour remise à zéro. A mon avis, réduire le taille de "0", mettre "=" au bas droite, et créer un nouvelle icône "C" dans le place de "=" maintenant.

• Le résultat toujours arrive comme "xxx.xxx". 1+2 apparu comme "3.000". C'est le "%.3f" dans le *printf* dans l'haute de cet colonne. Jour avec!

• En peut tuer l'appli avec 1÷0! Vous avez deux options. Cherche pour infos sur "signal.h" et le gestion des signales d'erreur, ou – plus facilement – dans *maths_op*, si l'opération est un division, et *thisnum* est zéro, refus!

• Regarder mieux les choses trop gros. Par exemple, 1234567890×1024 apparu comme 12641975193. SciCalc dit 1.264197519E12 qui est un moyen geek pour dire 1264197519000 (je crois?). Certainement, un gros chiffre avec 10 chiffres multiplie par un peu plus d'un mille, nous anticipe un résultat avec autour de 13 chiffres. En peut dire le moyen plus facile est de regarder tout réponse avec plus que ~10 chiffres comme "Trop large!".

Pour jouer avec cet application, on peut trouvera le code source sur: **http://www.riscos.fr/frob31/**
Bon courage!

¶

Mes chères françaises...

Je sais qu'il est un rêve de les français que le langue française deviendra le langue principale du monde, et tout le monde connaître le culture français.
À ce moment, *it is just expected that everybody speaks English*. Le langue des sciences, le langue principal a l'étranger, c'est presque toujours l'anglais.
Mais, un chose reste claire. Le monde anglophone perdre leurs appétits pour les affaires internationaux. L'élection de *Mr. Trump* est comme les américains ferme la porte dans un manier violent, puis construire un mur de béton derrière la porte. Peut-être physiquement, dans le cas de la frontière avec la Mexique.
Et, après *Brexit*, c'est évidemment que les anglais n'a pas d'appétit pour les étrangers (depuis le référendum, le violences aux étrangers augmente brutalement – autour de 70% **plus** que l'année précédent).

Voilà. C'est le direction de le principaux du monde anglophone à ce jour.

Pour la France? Mme Le Pen souhaite de suivre dans le même direction – sans doute un idée catastrophique pour le royaume uni, et qui touche aussi les américains. C'est pas mal, le rhétorique. La France comme dans les années soixante. Mais, attendre. C'est pas vrai. Considère, moi, j'aime bien les années 80. C'est ma jeunesse quand je suis un ado, c'est bon musique, c'est bon temps, tout va bien...
...actuellement, c'est l'époque de Margaret Thatcher, la guerre dans les îles Falklands, un Allemagne dévissé, plusieurs testes des bombes nucléaires, et aussi le Cold War entre US et la Russe.
Je souviens le choses que je souviens, et comme ado, les choses que je décrire au dessus fait pas trop d'intérêt pour moi. Les mémoires et la réalité, pas de tout le même chose.

En plus, c'est pas de tout le même monde. C'est une monde plus globalisé, une économie globale, ou la France fait commerce avec toutes les pays.

Donc, permets moi, s'il vous plaît, d'offrir un autre perspective.

La France. Déjà un pays puissante. Déjà un pays avec liens dans le monde musulman (peut-être un chose important dans l'avenir). Déjà un pays avec le deuxième langue diplomatique.

Pour pousse, pour deviendra encore plus important dans la monde, dans les affaires du monde, et pour remplir le trou qui est laissé par le monde anglophone... *c'est possible.*

La vérité, le langue et culture de France ne replace pas le langue et culture anglais/américain dans un seul nuit. Mais, si on votera pour les idées protectionniste, il jamais arrive. Jamais.

Trouve ça qui passe dans le monde aujourd'hui d'être pas quelque chose d'avoir peur. C'est une opportunité.

Si les anglais souhait de quitter l'Europe (peut-être complètement, personne ne sais rien sauf pour le première ministre – elle est un raté avec des exigences irréalistes), et les américains mettre en place un président qui fait politique par Twitter et qui changes ses idées plusieurs fois dans le même entretien (et, apparement, pense que la vérité est un chose optionnel)... ce qui, exactement, qui aide de gérer le paix, qui aide les pays démunis, qui aide de résoudre les problèmes internationaux? La Russe? Les Chinois? Pourquoi pas La France, un pays fondamentaux dans le union européen? C'est une opportunité.
Le monde sont géré pour longtemps par les anglophones. Ils, qui, maintenant, qui ont fatiqué du monde...ou plus pire, qui ont peur du monde. Pour raisons quasi-raciste et quasi-insulaire, ils ne souhaitera de continue comme avant.

Mon chère français, c'est pas un raison de fait même. C'est un opportunité. Un grand opportunité.
D'être audacieux.

¶

Ovation

Notre aventure commence en 1989 – l'année de le tombe du mur au Berlin, les premières pas sur l'internet (et l'initiale spécification pour le WWW nous prenons tous pour acquis), et aussi c'est un titre d'un album par Taylor Swift (elle est né en '89). Moi? Je suis au quatrième an dans mon école secondaire (lycée), comme internat.

Après je voir le puissance de BBC BASIC pour faire mes devoirs, mon parents m'achète un BBC Micro, deuxième main en 1987. Je l'utilise pour deux ans, et quand je vu le naissance de l'Archimedes, j'ai fait beaucoup de bruit pour une machine RISC OS. Finalement, un décision sont fait et voilà, c'est moi le propriétaire d'un *Acorn A3000*.

Avec un seul lecteur de disquettes, moniteur RVB à 50Hz, et un petite 1Mo RAM, il était probablement le moins puissant de la famille Archimedes. Mais en comparaison d'une BBC Micro, il était comme une Porsche contre une Cacahuète.

Je suis invité d'acheter un logiciel pour utilise avec la machine. Quelque chose que je souhaite depuis longtemps est un "Desktop Publisher", ou PAO en française.. J'écris un journal *pas* officiel à l'école (la plupart du temps je moquer du personnel!) avec EDWORD sur le BBC Micro. Alors, quand l'opportunité arrive, je demander le logiciel PAO de Acorn qui a été appelé ... *Acorn DTP*. Comme un nom, il est fonctionnellement descriptif, mais complètement dénué d'imagination. Plus ou moins, ce décrit aussi le logiciel. Il fait le travail, mais plutôt que vous aider, il semblait un désire d'être difficile. Entre vous et moi, j'ai un suspicion qu'AcornDTP était une démonstration de le nouveau système de police avec des contours, et par hasard il sont transformé en un produit pour vendre.

Comme même, il était des kilomètres d'avance en comparaison d'EDWORD. Actuellement, il était des kilomètres d'avance sur toute autre chose. Windows n'existé dans toute forme utile en 1989. C'est un monde des Ataris, des Amigas, et une douzaine de ordinateurs 8-bits que le temps bien oublié.
Je portera mon ordinateur à l'école, et le faculté me demander de le garder dans la salle des ordinateurs, ferme par clé, où "il serait en sécurité". Donc, je mettre mon A3000 sur un de les tables, entouré avec BBC Micros et Master Compacts. Comme il est une machine Acorn, il y un adaptateur Econet à l'intérieur. Je ne dit trop à les autres de le capacités de mon A3000, spécifiquement je ne dit rien pour le petit logiciel écrit en assembleur qui était capable d'enregistrer tous les donnes Econet dans un grand tampon dans le mémoire. Après un peu de trivial, il peut reconnaître les transferts et discarder les qui est inutile. Et après? C'est vraiment trop facile de recueillir tous les mots de passe. Econet n'utilise aucun façon de cryptage et tous les données apparaît sur chacun terminal branchée au réseau. Génial, hien?

Je pense que mon A3000 reste sur cet salle pour environ une semaine avant que quelqu'un volé tous les disquettes que je gardais avec elle. Mon premier logiciel en assembleur sont perdu, mais pas de grand chose, il servir son but – je sais tous les mots de passe, espécialement pour l'administrateur. Mais, un plus grande problème, AcornDTP sont aussi perdu.

J'ai un grande discussion avec l'école, qui refus de me rembourser. Je suis le seul qui est autorisé de entrer le salle des ordinateurs tout seul (originalement pour crochetage (!), mais après mon ordi arrive, ils me donne un clé). À tout autre temps, en peut anticiper un professeur d'être présent dans le salle.

En vérité, le professeur est un flocon spéciale (Google pour "special snowflake" en anglais!), qui souvent ouvrir la porte, puis laisse pour chercher les trucs pour le leçon, avec tout le monde dans le salle sans surveillance.

Enfin l'école me paye le prix de le logiciel volé.

Par cet temps, j'entends de *Ovation*, qui est récemment mettre en vente. J'achète ça, et vraiment, je utilise Ovation pour environ cinq minutes avant de décider que le vol d'AcornDTP sont tout à faite le meilleur destin. J'imagine le disquette à était formaté pour un Atari ou pareil. C'est un bonne finition pour AcornDTP qui sont vraiment absolument horrible.

Ovation? Je dit quoi, moi? Un éditeur WYSIWYG, d'entrée direct de texte, les menus sympa? Le différence entre Ovation et AcornDTP est comme le différence entre AcornDTP et EDWORD. Je gen qu'écrit Ovation actuellement anticiper des gens d'utiliser son logiciel pour créer les documents. C'est incroyable, aussi, d'avoir un logiciel PAO complète capable de fonctionner sur un machine de 1Mo avec des disquettes (mais 2Mo si on utilise aussi le correcteur orthographique).

Après cet incident, je sortir mon ordi de cet salle et mettre dans le dortoir. Nous sont le dortoir le plus geek. Vers le gauche de mon A3000, un ZX Spectrum 128 (avec un lecteur des cassettes intégrée), et vers le droite un Mac Classic (avec un petit écran monochrome et un lecteur disquettes qui s'éjecter les disquettes automatiquement! Cool!).

Et, vous sais quoi?

Personne toucher mon ordinateur. *Personne*. Rien volé. Et mieux, je peut tomber du lit au matin et allume l'ordi avant que j'ouvre mes deux yeux!

J'écrit beaucoup des choses avec Ovation. Fiches d'information, annonces (dans un magazine nationale, PDF n'existe pas et il n'accepte pas PostScript, donc je imprimer sur un LaserJet et ils scanné pour leur PAO avant d'impression), et des dizaines des histories merde jamais finis (le suis bon avec les débuts, mais le part après sont toujours un problème!).

Franchement, sauf pour SMPlayer/VLC je penser que Ovation est le logiciel que j'utilise le plus. Originalement Ovation, et ces jours, OvationPro (voilà, j'ecris cet magazine sur OvationPro sous RISC OS).

Je utilise Ovation(Pro) moins maintenant parce que l'Internet. Je peut rester comme un zombie et regarder Madoka...
Voilà. C'est pourquoi quand David Pilling publie le code source pour Ovation, je sauter à son site web. Originalement c'est uniquement pour l'intérêt – comment fonctionne un logiciel PAO? Mais au même temps, je sais que Ovation à des problèmes avec les machines modernes, sans doute à cause de son âge. Le logiciel arrive à l'âge de 27 ans. C'est plus qu'un éternité pour logiciel!

Je fait un décision, simplement pour le plaisir et la nostalgie, de modifier Ovation de fonctionner sur le Raspberry Pi et la nouvelle génération de machines RISC OS. J'ai aucune raison réelle, je ne l'utilise Ovation plus, après avoir passé à son successeur OvationPro. Cependant, il a été un ami fiable pour une grande partie de ma vie (y compris quelques fois, je l'habitude d'écrire des histoires science-fiction vraiment épouvantables, peut-être comme un moyens d'échapper une réalité que j'aurais préféré être arrivé à quelqu'un d'autre). Je sentais que je le devais.

Le travail du conversion Ovation vers 32 bits sont déjà effectuée. La plupart d'Ovation est écrit en C avec seulement de très petites pièces en assembleur, et pour ça, sauf pour quelques petits trucs, la conversion demande rien de plus que simplement passe tous vers le compilateur pour généré un exécutable compatible.

Bien sûr, c'est ici ou nous trouvons le premier obstacle. L'environnement de construction de David Pilling est un ensemble des scriptes aux ligne de commande extrêmement arcane, qui ont fait dans un âge avant les outils DDE et la capacité de construire des logiciels de manière GUI. David décrire le processus de construction sur son site, mais je ne voir pas! Meuh! Donc, je tout simplement jeté tout de ces choses et arrangé les fichiers comme une application DDE et écrit un Makefile pour effectuer la construction quasi-automatique. J'ai aussi simplifié les options – aujourd'hui il n'y a aucun raison de créer un version sans correcteur orthographe, et il aussi il n'y a pas de raison de créer un version de démonstration.
Les avantages? Je peux commencer la compilation du logiciel par appuyer sur Shift-Ctrl-C dans Zap directement à partir d'une source que j'ai modifié. Et le système fonctionne avec Throwback, donc si quelque chose va mal, je découvrir pas seulement quoi, mais aussi où, et je peux avoir le fichier ouvert à l'endroit correct. C'est des petites choses comme ça qui rendent la programmation beaucoup plus agréable.
Pour mettre Ovation sur mon Pi, c'est un joli simplicité.

Malheureusement, les versions récentes de RISC OS introduit un nouvel obstacle, qui montrer le bouchon. Le "*mémoire de page zéro protégée*" où tous les vecteurs de processeur et les données système importantes vit ailleurs (vers l'adresse &FFFF0000)

qui laisse rien a l'adresse &0. C'est pas dans le carte de mémoire, donc tout accès fait un défaut, un exception. C'est grave...

Ovation ne fait usage de cet mémoire, tous existe dans Ovation avec des appels système légales. Pourquoi est-ce un problème est quand on considère un tableau qui est pas initialisé. En général, il pointera à NULL, qui est une autre façon de dire zéro.

Quelque chose Ovation fait est comme:

```
if(
    (!s->box && s->page==act->page && !act->box)
    ||
    ( (s->box)->type==XMAIN && (s->box)->type==(act->box)->type &&
      shellof(s->box)==actshell
    )
    ||
    s->box==act->box
  )
```

Regarde bien. Le première ligne des comparaisons contient `!act->box` (on peut estimer que act->box est NULL là), et avec un comparaison OR (ou), le deuxième ligne contient un référence au `(act->box)->type`. Voilà, si `act->box` est NULL, ou pointera `(act->box)->type`?
Actuellement, il y beaucoup des petits parties ou Ovation simplement ne voir si un référence est NULL avant d'essayer de utiliser.
Mais tout, tout créer un accès mémoire au page zéro et si page zéro n'existe pas.....

Une exception est ce qui se passe. C'est est une façon polie que dire quand le programme marchera dans un mur, s'effondre sur le sol, puis vomis sur lui-même.
Comme je déjà dit. C'est grave.
C'est pas trop difficile de réparer. En prendre les comparaisons au dessus et sépare les, pour voir si le tableau est NULL ou non. Si non, nous peut progressons vers le reste des comparaisons avec connaissance que si le tableau est pas NULL, nous peut examiner un élément du cet tableau.

```
if ( ( act->box != NULL ) && ( s->box != NULL ) )
{
 if(
    (!s->box && s->page==act->page && !act->box)
    ||
    ( (s->box)->type==XMAIN && (s->box)->type==(act->box)->type &&
      shellof(s->box)==actshell
    )
    ||
    s->box==act->box
  ...
}
```

La plupart de chose que je changé dans Ovation est comme ça, bien que certains des bogues sont extrêmement compliqué, a résultat d'un pointeur NULL mis en place une demi-douzaine des appels de fonctions d'avant!

La prochaine série de problèmes à résoudre est relatives à l'âge du logiciel. Quand Ovation sont écrit, la plupart des gens utilisaient disquette et seul les riches utilisés les disques durs. Sérieusement, un disque dur de 20Mo (attention: c'est que des *mégaoctets*) vous coûtera le salaire d'un mois entier. Dix mille francs pour un disque qui contient seulement deux ou trois morceaux de musique en fichier MP3? Oui, bienvenue au années '80, comme démarrer votre disque dur fait raison pour les lumières dans le maison scintillant pour un petit moment.

Avec cet esprit, Ovation fait des suppositions comme les noms de fichiers est que dix caractères, et le nidification des dossiers fait un nom de fichier au moins de 128 caractères en total (le vrai limite est environ 230 caractères, mais 128 caractères ne sont pas une limite déraisonnable, si les fichiers sont cachées trop profondément, ils risquent de se perdre.
Aujourd'hui, Ovation refuse de charger un fichier si le complet nom de fichier est trop long. Je ne changera cet caractéristique parce que beaucoup trop de choses marchera avec cette hypothèse.
D'autre part, la plupart de mes noms des fichiers ces jours sont plus longs que dix caractères. Comme la plupart des cas de cette limitation est seulement un problème d'affichage, je tronque simplement des noms qui sont trop longues et le suffixe des points de suspension à la fin. "MonFichierExemplaire" deviendra "MonFichie…". Cet action arrête la corruption de mémoire évidente si Ovation essai d'ecrire un nom longue dans un espace pour seulement dix caractères.

Juste pour les *lulz*, je décide de mettre en oeuvre une measure de capacité pour Ovation de travailler directement avec Unicode.
En réalité, cet démarche sont extrêmement simple. Ovation cache des séquences de contrôle dans le

texte. Ces séquences fait des actions comme la modification du style de texte, la couleur, la police, et cetera. A cause de les séquences, le programme n'a pas faire le défectueux hypothèse fatale que un caractère à l'écran est égale à un caractère dans les données.
Au lieu de ces choses, Ovation balaye la ligne de texte de gauche à droite pour déterminer où est le point d'insertion précédent ou suivant. Avec cet comportement, c'est pas difficile de enseigner Ovation comment géré les séquences UTF-8. En fait, les codes UTF-8 contiennent leur longueur (en octets) à l'intérieur du codage lui-même. Les caractères UTF-8 peut demande entre 1 et 6 octets (en réalité, 4 octets sont les plus gros aujourd'hui mais 6 est la limite). Le octet initial, peut avoir bit 7 clair (pour ASCII, un seul octet) ou fixé pour un code UTF-8. Si bit 7 est fixé, cet bit et les fixés suivant donnera le nombre des octets. Il est suivi par un bit clair. Par exemple:

ASCII:	0xxxxxxx		
UTF-8 2 octets:	110xxxxx	xxxxxxxx	
UTF-8 3 octets:	1110xxxx	xxxxxxxx	xxxxxxxx

Pour mettre les caractères UTF-8 dans un texte, il demande absolument aucun travail. Rien. C'est toute un fonction du FontManager quand Ovation ouvrir les fonts avec codage UTF-8. FontManager peut dire "bonjour" même que 待っていたよ ou des autres langues. Il est juste des données qui référer des glyphes de caractères.
Plus importante, cet exercice est un essai de voit si il est pratique pour un application de utilise les caractères UTF-8 sans mettre le système entière dans l'alphabet UTF-8. Le réponse, oui.

Pourquoi? Parce-que il est le seul moyens réaliste pour des applications RISC OS de marche avec des caractères plus étrangers qui dans les caractères de ISO 8859/1 (le Latin1). Le problème, le plupart des gens qui travail avec ou sur RISC OS parle anglais. Pour ils, le question de changement vers UTF-8 sont facile – charger l'alphabet. Presque tout afficher pareil en anglais. Mais, c'est pas vrai pour tout le reste. Les accents en français comme é ô etc. Le ß en allemande... tout est corrompu par un changement vers UTF-8. C'est pas un solution. Il n'y aucun solution qui marche. Franchement, je pense que il est nécessaire de créer un Wimp qui peut marche en UTF-8 ou Latin1 au même temps. Mais, parce que les caractères utilisé en anglais reste sacro-saint, il n'existe aucun impulsion de change. Mais, alors, même UTF-8 est un compromis parce-que il est par pluparts les anglophones qui gérer les caractères dans le répertoire, le paresseuse stupidité qui est les caractères "CJK" parce-que Chine, Japon, et (ancien) Corée utilise les mêmes symboles, oui? C'est vrai qui l'origine de l'écriture est les caractères chinois, mais "CJK" est pour les asiatiques de l'est un peu comme si quelqu'un fait un "unification européen" et mélange les caractères latines avec l'alphabet cyrillique et un peu de grec...

Nоцs ρεцt écrιre commε ζα, ηοη?

La dernière chose a examiné est l'image et les cadres avec des couleurs déglingués. Encore une fois, c'est un résultat des changements dans la façon dont les choses fonctionnent aujourd'hui contre en 1990. À l'époque, il est un maximale de 256 couleurs sur l'écran au même temps. Un seul octet suffit. Ces jours? 32 mille, 64 mille, ou normalement 16 millions des couleurs. Un seul octet ne suffit pas quand l'encodage utilise un octet pour chaque de les trois couleurs primaires (rouge, vert, et bleu). J'enseigner Ovation que certains modes a besoin de deux ou quatre octets pour contenir les données de la palette. Quand ça a été fait, les bordures de cadres retournés à leurs couleurs correctes.

Je ne prévois pas de travail plus avec Ovation. Je essayer de corriger des bogues, mais le produit fonctionne bien maintenant sur les nouvelles machines. Je reçu un certain nombre de suggestions, mais la réponse à la plupart est *OvationPro*. Je ne change pas Ovation d'être OvationPro.

Ovation ... il est un peu daté, mais il fait toujours le travail et effectivement et bien. Il fait la mise en page simple, et couplé avec *PrintPDF* il est facile de créer de bons documents. Vous savez, je ne comprendre pas pas pourquoi Beebug ne donné pas un peu plus d'amour et d'attention a Ovation. Ovation pourrait ne pas avoir la vitesse brute parce qu'il est écrit en C, mais nous sommes toujours en attente d'une version 32 bits d'Impression. Vraiment, la décision de David a payé dans le long terme, car Ovation(Pro) fait non seulement à 32 bits, mais à une architecture tout à fait différente. Imaginez ce qui existe si Beebug pris Ovation plus sérieux.

Pendant longtemps, Ovation était un ami à moi.
J'espère, en apportant Ovation aux nouvelles machines, je suis été aussi un bon ami en retour.

¶

Le party du wrappage

Un traduction littérale de “wrap party” est “fête d’emballage”. C’est pas correcte. Google translate me dit que “wrap party” est... “wrap party”. Nous sont jamais sûr si “wrap party” en française est actuellement “wrap party” ou si Google juste ne comprendre pas l’expression.
Donc, un “wrap party” est un fête au fin de tournage quand tout les scènes ont filmé, les acteurs fameux peut aller fait autres choses, et le procès compliquée de post-production démarre (qui donne un explication de toutes les noms qui passe devant nos yeux quand le film est fini).

Et voici, cet édition spéciale de Frobnicate est fini. Devant votre yeux, trois noms –

David Feugey
qui me demande et donne l’impulsion de créer un dernière dernière (!) édition

Stephanie-Jane Murray
pour un peu de poésie en espagnole

Rick Murray
c’est moi, je créer le reste, et si le française ici sont vraiment penible, c’est ma faute...

Et, pour remplir toutes les autres noms –

Tout les crétins en angleterre qui voter Brexit.
quand le merde frappe le ventilateur,
c’est un peu trop tard de dit “oh, oups”.

Comme même, c’est un vrai exercice d’écrire un édition complet *en française*! Notamment parce que quand j’arrive en france en 2002 je peut dire pratiquement aucun française. Mes premières cinq ans, je ne parle pas trop avec des gens. Au travail, initialement je sais peu, j’améliorer dans le passage des temps, mais j’ai pas le type de personnalité de parler avec tout le monde. Je suis bien capable de passe un journée entière sans dire un seul mot. Mais c’est toujours obligé de dire “bonjour”, “à demain” et cetera. Je suis pas anti-social par intention, c’est normale pour les *introvertis*.

Mais, moi, j’écris. Beaucoup. Beaucoup de caca, bien sûr, mais beaucoup de choses. Pourquoi? C’est simple. Quand je parle, j’ai pas de temps pour préparation, pour réfléchir mes raisons et mes objectives. Je n’aime pas de parler. Mais pour écriture, c’est différent. J’écris un peu comme “stream of consciousness” (flux de la conscience) avec les phrases normale en langue parlant, parce que les mots écrits, c’est comment je parle mieux.

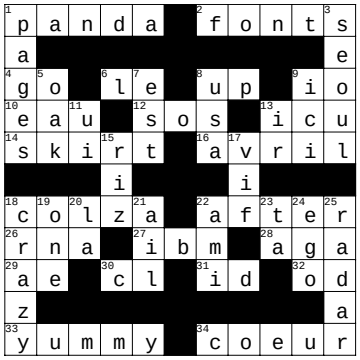
Et en française? Mon dieu! Je vous promi un chose: je n’écris pas en anglais et donnera tout au Google Translate. Je utilise le technologie pour chercher des bons mots. Mais, le plupart, c’est moi et un clavier.

Cet édition de Frobnicate prendre plusieurs soirs, weekends, et un ou deux weekends entières. C’est un chose important de créer un magazine entière dans un langue étranger quand on n’est pas sûr de leur compétence dans cet langue! Mais, c’est fait. Même s’il y un montagne des erreurs, je suis fier de cet petit magazine. Pourquoi? Parce-que j’essayer. C’est le chose le plus important. D’arrêt d’avoir les grandes idées et actuellement d’essayer de fait les choses.
Voici, Frobnicate, édition trente-et-un.
Et pour toutes mes fautes, je suis désolé.

Fait accompli? Non, pas complètement.

Pour vous, mes jolis lecteurs – ***merci***. Chaque personne qui aime ou apprendre quelque chose dans les pages, c’est un vie un petit peu différent grâce à mes efforts. Et, dans ces moments, vous peut me dire: *fait accompli.*

Merci.



Rick, 2017/02/19 à 23h50.