Tricks pour mon ordinateur

Thierry « tTh » Boudet

12 janvier 2024

Chapitre 1

Introduction

Des notes informelles sur l'utilisation d'un ordinateur moderne, vous allez en avoir vraiment beaucoup. Mais elles sont en vrac, et surtout, elles sont reprises d'une très ancienne version (circa 2011) et donc, peut-être pas vraiment d'actualité.

Cependant, toute cette ouvrage est en cours de réécriture avec Mousepad et LaTeX selon la méthode éprouvée dite de « La Rache », essentiellement sur une machine Debian/amd64, et peut-être de la Slackware qui traine.

Mes sources sont assez diverses [1]. Quand j'ai un souci précis à régler dans l'urgence, je commence par les *manpages* qui contiennent souvent la bonne piste, éventuellement avec l'installation d'un nouveau paquet plus approprié. Ensuite, je passe pas mal de temps à téléglander, et je tombe parfois sur de petites perles que je m'empresse d'incruster dans un chapitre. Et pour finir, il y a quelques ouvrages de référence. Citons, par exemple, le K&R, ceux de Christophe Blaess, le Christian Pelissier, et bien d'autres.

Le code LaTeX et les fichiers d'exemples sont disponibles en ligne sur le site du Tetalab. Your mileage may vary...

`https://git.tetalab.org/tTh/TetaTricks`

1. Il m'arrive même de parfois les citer, si j'y pense.

Table des matières

Chapitre 2

Le Shell

Le `shell` est le coquillage qui isole le MC/P [1] des yusers. En gros, hein, je vais simplifier, et me limiter aux bricolages constructifs et/ou amusant. Il y aura du gore aussi, parce que le shell est à la fois un interpréteur de commande et un langage de programmation. Ces deux notions ont beaucoup de choses en commun, comme nous allons essayer de le voir. Certains qualifient le shell de « langage de glue », et c'est une image assez pertinente.

Nous allons nous concentrer sur le *Bourne again shell* (aka `bash` [2]) qui est un des plus répandus, du moins dans le monde Linux. Mais il en existe d'autres : sh, ksh, ash, zsh, tthsh... Ce qui, bien entendu, pose quelques problèmes de compatibilité.

Ou alors, il faut reste POSIX, mais c'est pour le moment une autre histoire. Parce que là, on est plutôt dans une optique « morefun » sur l'utilisation de nos machines.

2.1 Interpréteur

Ceci est un prompt : `tth@redlady:~$`. Son petit nom est `PS1`. On peut y mettre plein d'informations utiles et de *mises en forme* assez funky [3] permettant d'enhancer la reliability de votre UI/UX. Mais surtout il est là pour indiquer qu'il attend vos ordres, qu'il est prêt à interpréter vos lignes de commandes.

```
tth@redlady:~$ date --utc
Thu Oct  5 00:53:01 UTC 2023
tth@redlady:~$
```

2.2 Langage

Le shell est aussi, et surtout, un langage de programmation, parfois appelé *glue-language* parce qu'il permet de "coller" ensemble plusieurs opérations élé-

1. Master Control Program
2. aka GNU/shell ?
3. Un peu de calme, fatalerrors !

mentaires, d'abord avec ses opérateurs internes et ensuite l'utilisation d'outils extérieurs [4] divers. C'est *très* efficace.

2.2.1 Le SHEBANG

Le shebang, c'est ce couple de caractères `#!` qui apparait au tout début d'un script shell. Le 'she' est l'abréviation de 'sharp', coupant en français. C'est probablement une allusion aux cotés tranchants du `#`. Quand au 'bang', c'est le point d'exclamation après l'explosion de la forkbomb.

Quand on demande au noyau d'exécuter un fichier (`man 3 exec`), il va en lire d'abord les premiers octets. Si les deux premiers sont *0x23 0x21*, le reste de la première ligne sera interprété pour lancer un autre exécutable, avec un traitement particuler des options.

2.3 Le pipeline

Une invention géniale, dont vous pouvez voir un exemple d'utilisation par un gadget sonore en page 70 et un bout de code C en page 20.

Exemple rapide, où la commande `uptime` envoie sa sortie dans le tuyau, et la commande `tr` va lire le tuyau pour son entrée :

```
$ uptime | tr ' ' 'X'
X09:22:29XupX33Xdays,X14:00,XX8Xusers,XXloadXaverage:X0.12,X0.17,X0.17
```

Bien entendu, on peut utiliser plusieurs tuyaux pour construire une chaine de traitements [5]. On peut même « intercepter » ce qui transite par un des tuyaux.

```
$
```

2.4 Structures de contrôle

2.4.1 Les tests

Deux éléments sont ils-égaux ?

2.4.2 Les boucles

2.5 Les fonctions

Une bonne manière de rendre votre code modulaire, et de réutiliser des bouts de code à volonté depuis plusieurs programmes.

Voici quelques fonctions :

4. Grep, Sed, Awk, BwBasic...
5. La chaine de compilation EQN/TBL/ROFF est un bon exemple

```
#!/bin/sh
message ()
{
echo $1 | boxes
}
```

Et comment les utiliser :

```
#!/bin/sh
. ./fonctions.sh
message "hello world"
```

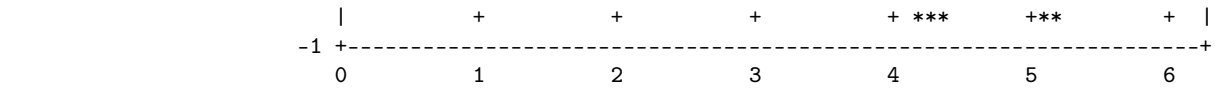
2.6 Heredoc

Un nom bien difficile à traduire, peut-être par « document en place » ?

```
#!/bin/bash

TITLE="Here Doc"

gnuplot << __EOC__
set term dumb
set grid
set title "$TITLE"
plot [x=0:pi*2] sin(x)
__EOC__
```

```
tth@redlady:~/Devel/TetaTricks$ code/shell/heredoc.sh
```

```
                                Here Doc

   1 +------------------------------------------------------------------+
     |          +**       ***         +          +          +          +  |
 0.8 |-+.......**.........:.**.......:.........:.........sin(x).*******-|
     |       ** :          :   *      :          :          :          :  |
 0.6 |-+....**..:.........:....**....:.........:.........:.........:+-|
     |     *    :          :      *   :          :          :          :  |
 0.4 |-+.**.....:.........:.......*..:.........:.........:.........:+-|
     |  *       :          :        * :          :          :          :  |
 0.2 |-*........:.........:.........**.........:.........:.........:+-|
     |*         :          :          *         :          :          :  |
   0 |-+........:.........:.........:**........:.........:.........:+-|
     |          :          :          :  *       :          :          : *|
-0.2 |-+........:.........:.........:..**......:.........:.........:*-|
     |          :          :          :    *     :          :          *  |
-0.4 |-+........:.........:.........:.....*....:.........:........**:+-|
     |          :          :          :      *   :          :       *  :  |
-0.6 |-+........:.........:.........:.......**.:.........:.....**...:+-|
     |          :          :          :         *:          :    **    :  |
-0.8 |-+........:.........:.........:..........**.........:..**......:+-|
```



Je pense que l'exemple est assez parlant. Pour un exemple plus complet, toujours avec gnuplot, c'est vers la page 84.

2.7 Les *locales*

Quelle sont les différences entre `en_US.UTF-8` et `C.UTF-8` ?

2.8 Questions à voir

- le traitement des chaines de caractères...
- la gestion des co-routines...

Chapitre 3

Le langage C

Un chapitre un peu particulier, puisque c'est le début d'une initiation au langage C pour les non-codeurs qui souhaitent rentrer dans le sujet à la dure.

Certaines des explications qui vont suivre ne sont pas très rigoureuses, mais montrent bien les principes généraux et les erreurs classiques. La futilité des exemples est assumée.

Les détails nécessaires seront présentés dans la suite.

3.1 Hello World

Hop, on y va...

Le tout début de la vie d'un programme écrit en C est l'appel par le *runtime*[1] d'une fonction nommée `main` qui recevra deux paramètres[2] dont l'utilisation est décrite un peu plus bas. Ces paramètres sont fournis par des mécanismes planquées dans la soute du système d'exploitation, et n'ont pas d'importance pour le moment..

```
#include  <stdio.h>

int main(int argc, char *argv[])
{
puts("hello world.");
return 0;
}
```

Un fois passé l'entrée, nous sommes dans la partie active. Nous appelons à ce moment une fonction de la bibliothèque standard : `puts` qui a pour but d'afficher sur l'écran le texte passé en paramètre.

Voila, c'est fait. Dans les règles de l'art. Nous savons dire « bonjour » au monde. Point suivant : dire bonjour à quelqu'un d'autre. Et pour cela il nous fait récupérer un argument depuis la ligne de commande.

1. `crt0.s`
2. En fait trois, mais restons simple.

3.2 Arguments

Expliquons maintenant les deux paramètres `argc` et `argv` du point d'entrée du programme (la fonction main). Le premier est le nombre de "mots" détectés par l'interpréteur de commande qui va lancer votre proggy, et le second est un tableau de chaines de caractères contenant ces différents mots.

Ce petit bout de code va nous afficher tout ça :

```
/*
 *      afficher les arguments.
 */

#include  <stdio.h>

int main(int argc, char *argv[])
{
int    foo;

for (foo=0; foo<argc; foo++) {
        printf("␣␣%3d␣␣␣␣%s\n", foo, argv[foo]);
        }

return 0;
}
```

Et voici un exemple d'exécution depuis un shell :

```
$ ./arguments un deux "trois quatre"
    0     ./arguments
    1     un
    2     deux
    3     trois quatre
$
```

Nous constatons que la première valeur affichée est en fait le nom de notre programme, ou plutôt le chemin vers le fichier exécutable, et surtout que son indice est 0, ce qui semble logique [3]. En C, les tableaux commencent toujours à l'indice 0.

Pour le traitement des options, il faut sauter à la page 24.

3.3 Les variables

En gros, une variable du C est une zone de mémoire destinée à contenir une valeur. Une variable peut être caractérisée par trois choses : son nom, son type, et sa portée.

Le nom : Il doit commencer par une lettre [4] majuscule ou minuscule, laquelle peut être suivie d'un nombre suffisant de lettres, de chiffres et du caractère 'souligné'. La différence de casse est signifiante.

3. Rez-de-chaussée, premier étage, toussa...
4. To be ASCII or not to be ?

Le type : C'est une désignation du genre d'information que l'on peut stocker dans une variable.

La portée :

3.4 Les fonctions

Nous avons vu brièvement dans la première section de ce chapitre la fonction `main` et ses deux paramètres. Il est temps de préciser les détails. Une fonction a un unique point d'entrée [5], avec une éventuelle liste de paramètres, et une tout aussi éventuelel unique valeur de retour.

XXX

3.5 Entrées / Sorties

Il est évident qu'un logiciel qui n'a aucun moyen de communiquer avec le monde extérieur a bien moins d'utilité qu'un bon ed. Dans le schéma « classique », ces échanges ont lieu par le truchement d'un terminal.

3.5.1 Les trois flux canoniques

`stdin`, `stdout` et `stderr`...

Par défaut, au lancement du programme, ces trois canaux d'entrée/sortie sont pré-connectés, et donc directement utilisables. Le premier (`stdin`), l'entrée standard, est connecté au clavier du consoliste [6], le second (`stdout`), la sortie standard, permet d'afficher les résultats du programme sur l'écran, et le troisème (), la sortie d'erreur, permet d'afficher les éventuels messages d'erreur. La différence entre *out* et *err* est expliquée plus loin.

3.5.2 IO de base

Les fonctions `getchar` et `putchar` sont les plus simples. Avec elles, on peut lire un caractère depuis `stdin`, et afficher un caractère dans `stdout`. Tout cela semble très bien, mais la fonction de lecture est piégeuse "IT'S A TRAP", comme nous allons bientôt le voir.

Vous devez logiquement penser que si nous allons lire un caractère depuis l'entrée du programme, nous pouvons utiliser une variable de type `char`, puisque ce type est prévu pour stocker un caractère. Seulement, pour indiquer l'abscence de caractère, la fin du fichier, il nous faut une valeur *out of band* et donc getchar renvoit en fait un `int` dont une valeur particulière sera nommée `EOF`, *end of file*.

```
while (EOF != (foo=getchar())) { ....
```

Voilà un piège éliminé, et vous trouverez un exemple complet un peu plus loin.

5. Qui a déja joué avec le ENTRY du Fortran ?
6. De quel roman est tiré ce terme ?

3.5.3 Écrire : `printf`

La fonction `printf` permet d'afficher sur la sortie standard le contenu d'une (ou plusieurs) variable sous diverses présentations, qui sont contrôlables par un tout petit DSL[7] : la chaine de format.

```
int bar, foo = 42;
bar = printf ("foo = %d\n", foo);
```

La première ligne déclare deux variables dont une (foo) est initialisée à une valeur connue. La seconde ligne appelle la fonction '`printf`' avec les paramêtres appropriés et conserve la valeur de retour de celle-ci.

Le premier de ces paramêtres, `"foo = %d\n"`, est appelé *chaine de format*. C'est une chaine de caractères, que l'on peut classer en trois catégories :

- *caractères crus* : ils sont directements poussés vers la sortie. La plupart d'entre eux donnent le résultat attendu.
- *codes de format* : les fragments qui commencent par le caractère '%'. Ils servent à contrôler l'apparence de ce qui va suivre. Par exemple `%x` va afficher un `int` en hexadécimal.
- *échappements* : les caractères précédés d'un '\' ou *backslash*[8] permettent d'inclure dans la chaine de format des caractères non imprimables. Par exemple, `'\n'` va générer le caractère *newline* qui marque la fin d'une ligne de texte.

Quand à la valeur de retour, elle contient le nombre de caractères effectivement écrits. Ce nombre peut être différent du nombre attendu en cas d'erreur, par exemple si le disque est saturé.

Ce nombre, que beaucoup de gens négligent, peut être utilisé pour un affichage de multiples données en limitant la longueur des lignes :

```
int foo, nbre = 0;
for (foo=0; foo<1000; foo++) {
        nbre += printf("%d ", foo);
        if (nbre > 62) {
                putchar('\n'); nbre = 0;
                }
        }
```

3.5.4 Lire : `fgets`

Cette fonction a pour but de lire une ligne de texte depuis une entrée (stdin, fichier, socket...) et de la ranger en mémoire. Mais commençons par lire la documentation de cette fonction, en nous concentrant sur ces deux passages, avant de passer à un exemple[9].

Voici deux extraits de ce que nous affirme la page du manuel :

7. *domain specific language*
8. barre-penchée-du-8
9. encore en phase de méditation

```
DESCRIPTION
   fgets()  reads in at most one less than size characters from stream and
   stores them into the buffer pointed to by s.  Reading  stops  after  an
   EOF  or a newline.  If a newline is read, it is stored into the buffer.
   A terminating null byte ('\0') is stored after the  last  character  in
   the buffer.

RETURN VALUE
   fgets() returns s on success, and NULL on error or when end of file oc-
   curs while no characters have been read.
```

Prenez le temps d'y réfléchir, et essayons un cas simple d'utilisation pour examiner le comportement de cette fonction. Nous allons boucler sur la lecture d'une ligne depuis l'entrée standard dans un buffer d'une taille fixée, puis afficher le nombre de caractères que nous avons reçus.

```
/*
 *               exemple d'utilisation de 'fgets'
 */
#include  <stdio.h>
#include  <string.h>
#define          TL      12
int main(int argc, char *argv[])
{
char             buffer[TL+1];
while( fgets(buffer, TL, stdin) ) {
        printf("␣␣␣%4ld␣␣␣␣␣␣%s\n", strlen(buffer), buffer);
        }
return 0;
}
```

3.5.5 Lire : `scanf`

Avez-vous bien révisé la section qui cause des pointeurs ? Êtes-vous prèt à d'étranges comportements ? À ne plus savoir où peut être la tête de lecture ? Alors les subtilités de `scanf` vont vous plaire.

To be continued...

3.5.6 Les fichiers

Lire et écrire depuis les flux gérés par le système, c'est bien, mais c'est mieux de pouvoir faire la même chose depuis ou vers un fichier enregistré. Il nous faut un moyen pour se 'connecter' à un ficher, et y balancer des trucs à grand coup de printf.

Ce mécanisme passe par la fonction `fopen` et `fclose`. La première va nous renvoyer (si tout se passe bien) un pointeur sur une structure opaque de type `FILE *` qui pourra être utilisé dans la suite des opérations. Voici son prototype :

```
#include  <stdio.h>
FILE *fopen(const char *pathname, const char *mode);
```

Le premier paramêtre est le nom du fichier concerné, possiblement avec son chemin d'accès (absolu ou relatif), comme `"foo.fimg"`, `"/var/tmp/foo.data"` ou `"./bla.txt"`. Le second est le mode d'accès à ce fichier. Ce mode précise, entre autres, si nous souhaitons lire ou écrire dans ce fichier. La valeur retournée par cette fonction est un pointeur qui peut être égal à `NULL` en cas d'erreur : par exemple le fichier n'existe pas ou ne peut pas être crée.

Notre fichier est maintenant ouvert, la fonction `fprintf` va nous permettre d'écrire quelque chose dedans. Elle est analogue au `prinf` vu un peu plus tôt, sauf qu'elle demande un argument supplémentaire, et c'est justement le pointeur retourné par `fopen` :

```
int fprintf(FILE *stream, const char *format, ...);
```

3.6 Quelques filtres Unix

Ce qu'on appelle un "filtre" est un logiciel destiné à transformer les données qui le traversent. Pour bien comprendre, le mieux, comme d'habitude, est un exemple. Il est un peu artificiel, car il s'agit d'éliminer les lettres O et P, sans raison valable. Le voici :

```
/*
 *      no-op.c is an useless shell filter
 */
#include  <stdio.h>
#include  <ctype.h>

int main(int argc, char *argv[])
{
int     quux, baz;

while (EOF != (quux=getchar())) {
        baz = toupper(quux);
        if ('O'==baz || 'P'==baz) continue;
        putchar(quux);
        }

return 0;
}
```

Démonstration immédiate :

```
$ echo apopoz | ./no-op
az
$ ./no-op < no-op.c | head -5
/*
 * n-.c is an useless shell filter
 */
#include  <stdi.h>
#include  <ctye.h>
$
```

Vous voulez un autre exemple ? En voici un, parfois utile pour des simulations de terminal de *tipiak*[10] sur une ligne en boucle de courant à 300 Bauds, le tout pour un film que l'on souhaite classable en série Z.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <unistd.h>

int main(int argc, char *argv[])
{
int             c, sec, nano;
struct timespec ts;
int             slowdown = 300;    /* set a default value */
float           delay;
/*
 *      get the desired slowdown from command-line
 */
if (2 == argc) {
        if (1 != sscanf(argv[1], "%d", &slowdown)) {
                fprintf(stderr, "%s:␣bad␣arg,␣sorry\n",
                                  argv[0]);
                exit(1);
                }
        if (slowdown < 1) {
                fprintf(stderr, "%d␣is␣invalid\n",
                                  slowdown);
                exit(1);
                }
        }

/* convert the slowdown value to delay */
delay = 1.0 / (float)slowdown;
sec  = (int)delay;
nano = (int)(1000*1000*1000*(delay - (int)delay));

ts.tv_sec  = sec;
ts.tv_nsec = nano;
while ( EOF != (c=getchar()) )
        {
        putchar(c);
        fflush(stdout);
        nanosleep(&ts, NULL);
        }

return 0;
}
```

Vous prendrez bien soin de contempler le traitement d'un éventuel paramètre passé par la ligne de commande, et surtout le calcul alambiqué[11] pour convertir une vitesse en Bauds à un délay acceptable par `nanosleep(2)`.

10. Le texte vert ou orange sur fond noir est de rigueur.
11. On doit quand même pouvoir faire plus clair...

Pour une explication plus détaillée sur les mécanismes utilisés et les usages possibles d'un tel filtre, il faut voir le concept de pipeline du shell en page 12.

3.7 Les pointeurs

Ah, enfin, on a failli attendre !

Qu'est-ce qu'un pointeur ? La réponse est multiple, et c'est le pointeur qui fait à la fois la force et la faiblesse du C. Pour faire simple, un pointeur est une variable qui contient l'adresse dans la mémoire d'une autre variable.

Mais en fait c'est un peu plus subtil : un pointeur « connait » la taille de l'objet pointé, et peut donc « parcourir » un tableau de ces objets.

```
Objet        tableau[N];
Objet        *ptr;
ptr = tableau;             //   *ptr désigne tableau[0];
ptr++;                     //   *ptr désigne tableau[1];
```

XXX il y a beaucoup à dire, et de quoi bien rire *:)*

3.8 Le préprocesseur

Nous avons déja abordé de loin la directive `#include`, qui fait partie du préprocesseur, comme toutes les lignes de code commençant par le caractère #. Le concept de base, qu'il faut bien capter, est que le préprocesseur pratique des *substitutions de texte*. Pratiquement, il se passe ça :

```
#define    NUMID        1664
#define    FLAVOUR      "tisane"
printf("La %d c'est de la %s\n", NUMID, FLAVOUR);
```

Ces trois lignes de code (deux directives pour cpp et un appel classique de fonction) seront converties en une seule ligne de C.

```
printf("La %d c'est de la %s\n", 1664, "tisane");
```

C'est donc cette ligne qui sera ensuite passée au vrai compilateur pour être traduite en codes opératoires, dont l'exécution affichera une maxime dont la véracité demande à être vérifiée par l'expérimentation. Mais ça n'est pas fini.

Ce sympathique préproc' nous permet aussi de faire de la compilation conditinnelle, et ça, c'est cool parce que ça ouvre la porte à plein de choses. Par exemple, l'instrumentation du code afin de faciliter les tests et le debug.

```
...
#ifdef TRACEUR
fprintf(stderr, "pid %d was here.\n", getpid());
#endif
...
```

Et à la compilation, il vous faut passer l'option `-DTRACEUR` à Gcc pour que ce message de trace soit pris en compte.

3.8.1 Les macros

Une chose très fourbe dont voici un exemple ?

```
#define pixidx  (fi,x,y)  (((y)*fi->width)+(x))
#define getRpix (fi,x,y)  (fi->R[ pixidx(fi,(x),(y)) ])
#define getGpix (fi,x,y)  (fi->G[ pixidx(fi,(x),(y)) ])
#define getBpix (fi,x,y)  (fi->B[ pixidx(fi,(x),(y)) ])
```

Finalemant, ça n'est pas si compliqué que ça. Il suffit juste de savoir protéger les choses fragiles avec des parenthèses. Partout.

3.9 Unités de compilation

Jusque à maintenant, nous n'avons vu que des programmes dont le code source n'était que dans un seul fichier, ce qui devient vite ingérable pour un gros projet. C permet facilement de faire de la compilation séparée : chacun des fichiers source est compilé indépendament en un fichier *objet*, lesquels seront ensuite **liés** pour obtenir l'exécutable final.

Mais découper un gros logiciel en plusieurs fichiers source a d'autres avantages. Bien maitrisé, c'est une technique assez sereine. Elle permet d'isoler des données spécifiques qu'il serait bien plus lourd et périlleux d'utiliser d'une autre façon.

3.10 Les structures

Une structure est une sorte de boite dans laquelle on peut ranger plusieurs variables afin de les manipuler comme une seule entité. Les utilisateurs de 80 colonnes voient tout de suite de quoi il s'agit.

XXX trouver un exemple parlant et décalé...

3.11 Gestion de la mémoire

Nous avons déja entrevu la gestion « implicite » de la mémoire avec les variables locales. Il est temps de passer à une gestion explicite de celle-ci : les fonctions `malloc` et `free` sont là pour ça.

La première fonction demande au mc/p de nous préter une certaine quantité de mémoire, que nous pourront utiliser à notre guise. Et la seconde restitue cette zone mémoire au système sous-jacent.

Un rapide synopsis minimal d'utilisation :

```
int     foo, *ptr;
if (NULL==(ptr=malloc(sizeof(int)*NBITEMS)))      abort();
for (foo=0; foo<NBITEMS; foo++)                   ptr[foo] = rand();
do_something(ptr);
free(ptr);
```

Pour les besoins de la démo, nous avons deux variables, l'une est entière (`foo`) et l'autre, `ptr`, est un pointeur sur N entiers. Ce pointeur est initialisé sur la seconde ligne par un appel à la fonction `malloc(3)` avec en paramètre le nombre d'**octets** que nous voulons emprunter. Ce nombre est ici calculé en multipliant le nombre de case désiré par la taille de la case, une bonne occasion de découvrir l'opérateur `sizeof`...

La boucle `for` de la ligne suivante insère des données pertinentes dans notre tableau. Lesquelles données seront habilement traitées sur la ligne suivante par l'appel de cette fonction :

```
void do_something(int values[])
{
int       foo;
double    sum = 0.0;
for (foo=0; foo<NBITEMS; foo++)     sum += (double)values[foo];
printf("sum is %g\n", sum);
}
```

Et finalement, nous avons réussi à générer un *useless number*. Notre mission est terminée, nous rendons notre bloc de mémoire au mc/p avec `free(ptr);`.

3.12 Gérer les options

```
#include <unistd.h>
int getopt(int argc, char * const argv[], const char *optstring);
extern char *optarg;
extern int optind, opterr, optopt;
```

La page de man de getopt(3) contient des explications détaillées et un exemple simple d'utilisation.

3.13 Analyser une ligne de texte

Nous désirons implémenter une interface en ligne de commande (aka CLI) pour un enchainement simple d'intructions assez semblables. On peut aussi voir ça comme la création d'un *domain specific language* (aka DSL).

Pour être cohérent avec le shell, nous allons utiliser la même méthode de séparation des mots sur la ligne à décomposer : utiliser une liste pré-définie de séparateur de champs, comme le $IFS du shell.

Pour simplifier l'exemple, toutes les commandes susceptibles d'être tapées seront structurées de la même manière.

3.13.1 la fonction `strtok`

Cette fonction nous permet de « découper » notre ligne de commande en *tokens* par un ensemble de caractères de séparation appelé `delim` dans cet extrait du man.

```
#include <string.h>
char *strtok(char *str, const char *delim);

The  strtok()  function breaks a string into a sequence of zero or more
nonempty tokens.  On the first call  to  strtok(),  the  string  to  be
parsed should be specified in str.  In each subsequent call that should
parse the same string, str must be NULL.
```

Concrètement, nous allons utiliser deux séparateurs très classiques, l'espace et la tabulation `delim = "\t ";`. on peut ensuite aller regarder dans la mémoire ce qui s'y passe :

```
   --- chaine d'origine
66 6f 6f 20 20 20 62 61 72 09 71 75 75 78 20 77 69 7a 00
 f  o  o           b  a  r     q  u  u  x     w  i  z
   --- 1er strtok
66 6f 6f \0 20 20 62 61 72 09 71 75 75 78 20 77 69 7a 00
 f  o  o           b  a  r     q  u  u  x     w  i  z
   got [foo]
```

Donc, `strtok` a détecté le premier caractère de séparation, ici un espace, qui est juste après le premier mot de la ligne à analyser, et l'a remplacé par un '\0', le marqueur de fin de chaine. Le premier mot (foo) a bien été isolé.

on peut constater que la chaine de départ est modifiée, ce qui risque de causer des soucis, mais on en parlera plus tard.

3.13.2 Le programme complet

```
/*
 *              +-------------+
 *              | demo strtok |
 *              +-------------+
 */

#include   <stdio.h>
#include   <string.h>
#include   <ctype.h>
/*-------------------------------------------------------*/
void dump_line(char *titre, char *ligne, int combien)
{
int             foo;
unsigned char   *ptr;

printf("␣␣␣---␣%s\n", titre);
ptr = (unsigned char *)ligne;
for (foo=0; foo<combien; foo++) {
        if (ptr[foo])   printf("%02x␣", ptr[foo]);
        else            printf("\\0␣");
        }
puts("");
for (foo=0; foo<combien; foo++) {
        printf("␣%c␣", isprint(ptr[foo]) ? ptr[foo] : '␣');
```

```
        }
puts("");
}
/*-------------------------------------------------------*/
int main(int argc, char *argv[])
{
char            input[] = "foo␣␣␣bar\tquux␣wiz";
char            *cptr, *copie;
int             foo;

dump_line("original", input, sizeof(input));
copie = strdup(input);
dump_line("copie", copie, sizeof(input));
puts("");

cptr = strtok(copie, "␣\t");
dump_line("1er␣strtok", copie, sizeof(input));
printf("␣␣␣got␣[%s]\n", cptr);
puts("");

puts("␣␣␣two␣more␣strtok...");
cptr = strtok(NULL, "␣\t");
printf("␣␣␣got␣[%s]\n", cptr);
cptr = strtok(NULL, "␣\t");
printf("␣␣␣got␣[%s]\n", cptr);
dump_line("2eme␣&␣3me␣strtok", copie, sizeof(input));

foo = cptr - copie;             /* this is the hack :) */

printf("␣␣␣index␣=␣%d\n", foo);
printf("␣␣␣and␣we␣have␣[%s]␣:)\n", input+foo);
return 0;
}
```

3.14 Erreurs classiques

Le C est un langage parsemé de fosses, de chausse-trapes et de pièges fourbes et sournois. Sans parler des UBs, hein...

- Variables non initialisées.
- Sortir d'un tableau (par le haut ou par le bas).
- Dépassement de la taille d'une chaine.
- Libérer deux fois la mémoire.
- *file pointer* invalide.

3.15 Random

XXX

Un ordinateur est déterministe.

3.16 Debug

Des outils et astuces en page 136.

> splint is an annotation-assisted lightweight static checker. It is a tool for statically checking C programs for security vulnerabilities and coding mistakes. If additional effort is invested in adding annotations to programs, splint can perform stronger checking.

3.17 Legalize

"The two forms of conforming implementation are hosted and freestanding. A conforming hosted implementation shall accept any strictly conforming program. A conforming freestanding implementation shall accept any strictly conforming program in which the use of the features specified in the library clause (Clause 7) is confined to the contents of the standard headers <float.h> , <iso646.h> , <limits.h> , <stdalign.h> , <stdarg.h> , <stdbool.h> , <stddef.h> , <stdint.h> , and <stdnoreturn.h> . Additionally, a conforming freestanding implementation shall accept any strictly conforming program in which the use of the features specified in the header <string.h> , except the following functions : strdup , strndup , strcoll , strxfrm , strerror ." (4p6)

https ://devblogs.microsoft.com/oldnewthing/20230109-00/ ?p=107685

3.18 Dessiner

3.18.1 G2

> **g2** is a simple to use graphics library for 2D graphical applications written in *ANSI C*. This library provides a comprehensive set of functions for simultaneous generation of graphical output on different types of devices.

```
/*
 *               DESSINER AVEC G2
 */

#include  <stdio.h>

int main(int argc, char *argv[])
{

fprintf(stderr, "file␣%s␣compiled␣%s␣at␣%s\n",    \
                         __FILE__, __DATE__, __TIME__);


return 0;
}
```

3.19 La suite ?

Étudier les différents modêles de mémoire partagée.

3.20 Ailleurs dans cet ouvrage

Il y a plein d'autres exemples de code en C, sur des sujets divers comme Open Sound Control (page 48) ou libsndfile (page 30). Et pour les gens du système : l'utilisation des signaux (page 119), le chargement dynamique d'un *plug-in* (page 130).

Chapitre 4

Makefile

"make -C dir" tells "make" to change to the specified directory before doing anything else.

Chapitre 5

Son

Abordons maintenant un sujet complexe : le son en général, la musique (p. 36) et la synthèse de sons (p. 38).

5.1 ALSA

ALSA, *Advanced Linux Sound Architecture*, gère à peu près correctement les entrées/sorties sonores d'une machine Linux, aussi bien en audio qu'en MIDI [1].

Nous allons faire le tour de quelques outils associés. En premier, le « centre de contrôle », `alsamixer`, dont la qualitée de l'interface en Curses est époustouflante :

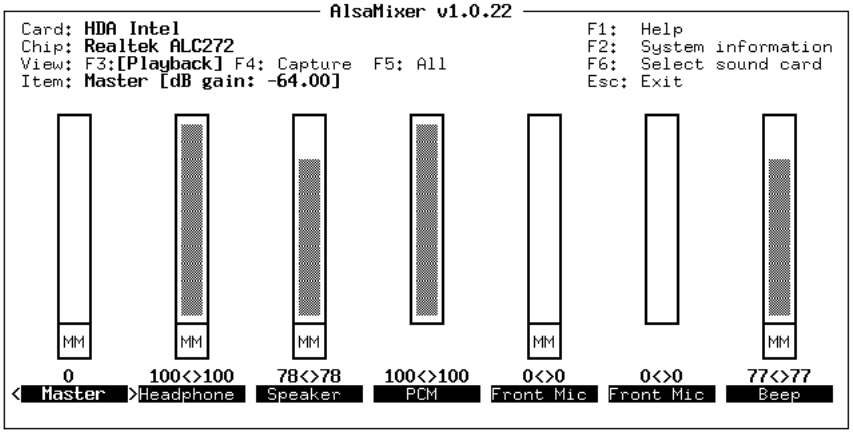


FIGURE 5.1 – Alsamixer

Ensuite, nous avons `aplay` et `arecord`, qui permettent d'écouter et d'enregistrer les sets de James Lewis [2].

1. Voir page 43 pour en savoir plus.
2. Que ça, c'est de la super noise !

5.1.1 arecord

Si vous avez plusieurs cartes son, la sélection du *device* est assez piégeuse : ce n'est pas l'option `-l` qui vous donnera la bonne piste, mais l'option `-L` qu'il faut utiliser.

```
arecord -D plughw:CARD=Snowball,DEV=0 -v -r 22050 foo.wav

arecord -D plughw:CARD=H4,DEV=0 -v -c 2 -f S16_LE -r 44100 foo.wav
```

5.1.2 Du code...

```
        int
       main()
      {float f
    ;char c;int
  d,o;while(scanf(
 "%d%c%d ",&d,&c,&o)
>0){c&=31;for(f=!(c>>4)*55,c=(c*8/5+8)%12+o*
                      12-24;c--;f*=1.0595);
                         for(d=16e3/d;d--;
                            putchar(d*f
                             *.032))
                               ;}}
```

https ://zserge.com/posts/etude-in-c/

https ://www.alsa-project.org/alsa-doc/alsa-lib/pcm.html

5.1.3 Le reste

Et quand on regarde par la lorgnette du programmeur, on découvre rapidement que l'utilisation d'ALSA n'est pas si simple que ça, essentiellement à cause d'une documentation spartiate. Heurensement, pour certains domaines, il existe des surcouches plus simple d'emploi : `libao` pour les sorties sonores, XXX pour le reste.

ALSA apporte aussi les primitives et des utilitaires de gestion des interfaces MIDI. Là aussi, la doc est sommaire. Quelques explications sur la page 43.

5.2 Configurer Alsa

TODO

5.3 pulseaudio

On avait pourtant dit « **pas de trolls** », mais il va quand même falloir y passer : ce machin prétenduement *michu compliant* a parfois, voire même souvent, tendance à se mettre en travers du chemin des honnêtes gens [3].

3. Du moins vers les années 2014/2019

5.4 Jack

Jack Audio Connection Kit : ce sous-système [4] est destiné avant tout aux professionnels de la musique. Je vais attendre le retour de mon spécialiste préféré pour lui laisser le soin d'en parler.

Jack peut aussi servir de source sonore pour darkice, lequel pourra feeder un serveur de stream comme Icecast (cf page 65).

5.5 libsndfile

Libsndfile est essentiellement une très complète bibliothèque de lecture et écriture de fichiers sons. Elle permet de gérer facilement un nombre considérable de formats de fichiers [5].

Elle arrive avec une floppée d'outils en ligne de commande, dont certains sont assez performants. Citons, entre autres, `sndfile-resample` qui est un convertisseur de fréquence d'échantillonage très performant [6].

5.6 libao

C'est une bibliothèque de fonctions pour gérer les sorties sonores d'une application (*Libao is a cross-platform audio library that allows programs to output audio using a simple API on a wide variety of platforms.*). Elle supporte une grande variété de formats.

```
/*
 * ao_example.c
 *      Written by Stan Seibert - July 2001
 *
 * Compilation command line (for Linux systems):
 *      gcc -o ao_example ao_example.c -lao -ldl -lm
 */

#include <stdio.h>
#include <string.h>
#include <ao/ao.h>
#include <math.h>

#define BUF_SIZE 4096

int main(int argc, char **argv)
{
ao_device *device;
ao_sample_format format;
int default_driver;
char *buffer;
int buf_size;
```

4. Dans le monde Linux, on aime bien les sous-systèmes pour le son.
5. Mais ceci est une autre histoire, que nous verrons peut-être un de ces jours...
6. Et probablement basé sur *Secret Rabbit Code*

```
int sample;
float freq = 440.0;
int i;

        /* -- Initialize -- */
fprintf(stderr, "libao example program\n");
ao_initialize();

        /* -- Setup for default driver -- */
default_driver = ao_default_driver_id();
memset(&format, 0, sizeof(format));
format.bits = 16;
format.channels = 2;
format.rate = 44100;
format.byte_format = AO_FMT_LITTLE;

        /* -- Open driver -- */
device = ao_open_live(default_driver, &format, NULL);
if (device == NULL) {
        fprintf(stderr, "Error opening device.\n");
        return 1;
}

        /* -- Play some stuff -- */
buf_size = format.bits/8 * format.channels * format.rate;
buffer = calloc(buf_size, sizeof(char));

for (i = 0; i < format.rate; i++) {
        sample = (int)(0.75 * 32768.0 *
              sin(2 * M_PI * freq * ((float) i/format.rate)));
        /* Put the same stuff in left and right channel */
        buffer[4*i] = buffer[4*i+2] = sample & 0xff;
        buffer[4*i+1] = buffer[4*i+3] = (sample >> 8) & 0xff;
        }
ao_play(device, buffer, buf_size);

/* -- Close and shutdown -- */
ao_close(device);
ao_shutdown();

return (0);
}
```

La configuration se fait par deux fichiers : `/etc/libao.conf` surchargeable par `$HOME/.libao.conf`, et schématiquement, nous pouvons voir qu'elle est à deux niveaux.

https ://www.xiph.org/ao/doc/drivers.html

5.6.1 Choisir la sortie

Pour choisr comment et vers quelle prise physique libao va envoyer le son, il y a deux choix à faire. La partie logicielle d'une part (aixs, oss, alsa, jack,

sndio...) et la partie matérielle (jack standard, casque audio USB...).

5.7 Portaudio

Une suggestion de Wargreen pour avancer un peu dans la construction de ma DAW[7] *custom-made* à mon idée...

5.8 Ladspa

Ladspa, mais c'est quoi ? C'est l'acronyme de « Linux Audio Developers Simple Plugin API », ou, en français : « machin avancé ».

La commande `listplugins`, comme son nom l'indique, fournit la liste de tous les plugins accessibles. Comment y acceder est une question dont la réponse est laissée à votre bonne volonté.

5.9 Sox

Sox est le « couteau suisse » des manipulateurs de données sonores. Sa documentation étant assez ésotérique, je vais faire comme d'hab' : présenter des exemples approximatifs.

5.9.1 Conversions

Nous avons un process quelconque qui nous délivre des données en format texte, tel que celles-ci, deux valeurs par ligne, normalisées dans l'intervalle $]-1,1[$, et nous voulons obtenir un .Wav vanille.

```
0.286498189155 0.33310689881
0.0500751550941 -0.326965910932
-0.01317994812 -0.392365740013
-0.353560435178 -0.018978716441
```

Voici comment faire dans un contexte de *Makefile* :

```
random.wav: random.dat Makefile
        @echo " conversion " $< $@
        sox --rate 44100 $< $@
```

5.9.2 Génération

```
play -n synth sine 480 sine 620 remix 1-2 fade 0 0.5 delay 0.5 repeat 5
play -r $[28*1024] -b 16 -c 2 -e signed-integer -t raw /dev/sda -S
play -r $[8*1024] -b 8 -c 1 -e unsigned-integer -t raw /dev/sda -S
```

7. Digital Audio Workstation, aka machine à sons

5.9.3 Traitements

`sox -norm[=dB-level] source.wav normal.wav` pour normaliser un son. *Automatically invoke the gain effect to guard against clipping and to normalise the audio.*

`sox left.wav right.wav -M stereo.wav` pour combiner deux voies mono en un fichier stéréo. `-M` est l'abréviation de `-combine merge`.

Dupliquer la piste d'un fichier mono pour obtenir un fichier stéréo : "`sox --t wav -b 16 - remix 1 1`" avec le premier tiret qui dit de prendre l'entrée standard

Questions en attente :

comment combiner huit pistes monophoniques en un seul fichier contenant ces 8 pistes ?

Quelles métriques Sox peut-il extraire ? Eg : *spectrogram*.

5.10 espeak

Logiciel de synthèse vocale. Mais c'est pas comme un spo256al2, hélas.

```
tth@plop:~/RADIO$ espeak --voices=fr
Pty Language Age/Gender VoiceName       File        Other Langs
 5  fr             M  french            fr
 7  fr             M  french-mbrola-1   mb/mb-fr1
 7  fr             F  french-mbrola-4   mb/mb-fr4
 5  fr-be          M  french (Belgium)  fr-be
tth@plop:~/RADIO$ cat hello.txt | espeak
```

Comme à l'accoutumée, nous allons chercher un contexte d'utilisation futile et réjouissant. Le texte qui sera « parlé » sera la sortie de la commande `ddate` [8], avec une voix d'outre-monde, sur un seul canal, ce que nous pouvons entendre avec "`echo $(ddate) | espeak-ng`".

5.11 ecasound

5.12 Pour finir...

En conclusion, le son avec Linux, c'est un peu la grande foire. On arrive parfois même à y voir du XML dedans et c'est en page 65 pour les courageux. On peut aussi carrément utiliser un langage spécialisé comme *liquidsoap*.

8. outil indispensable de la vie numérique `http://www.subgenius.com/`

Chapitre 6

Joueurs de musique

Vous pouvez aussi aller voir la page 30, qui traite d'autres problématiques liées au son.

6.1 mocp

Le nom « Mocp » est l'abréviation de Music on-consol Player. Personnellement, j'aime beaucoup son look, et son ergonomie parfois déroutante.

Les fichiers de configuration sont stockés dans le répertoire `$HOME/.moc/` et on y trouve la config en elle-même, et les thêmes [1]. Un exemple très complet (.../doc/moc/examples/config.example) est fourni. Les commentaires vous expliquerons bien tout. Voici un petit extrait de la configuration que j'utilise :

```
# enchainement des morceaux.
Repeat     =   yes
Shuffle    =   yes
AutoNext   =   yes

Theme      =   moca_theme
```

Pout un usage de base, c'est bien suffisant, mais il y a des options avancées qui permettent de faire bien d'autres choses.

6.2 mpd

Le nom « Mpd » est l'abréviation de Music Player Daemon. Il peut théoriquement [2] *feeder* un serveur Icecast.

1. Très importants, les thêmes de mocp :)
2. Théoriquement, parce que je n'ai pas encore essayé :)

6.3 Audacious

6.4 Effets sonores

Ladspa ?

6.5 Les playlists

Une *playlist* est (en général) un fichier contenant une liste de musiques à écouter.

Chapitre 7

Modulaires

Comment définir un synthétiseur modulaire en quelques mots ? Disons que c'est un synthé dont chaque composant élémntaire est indépendant, et interconnectable aux autres par un gros plat de cables sans parmesan.

Un univers intrigant et infini.

7.1 Chuck

Chuck peut être vu comme un équivalent logiciel à ce système de cablage. Nous avons sour la main des modules, que nous allons raccorder et paramétrer grace à l'opérateur `=>`. Le module `dac` correspond à la sortie audio de votre machine. Voici donc un premier exemple, sans la moindre prétention musicale :

```
// Just a little beep
SinOsc s    =>   dac;
0.5         =>   s.gain;
440         =>   s.freq;
2::second   =>   now;
```

Félicitation, vous venez de coder un diapason rudimentaire. La première ligne connecte un oscillateur sinusoïdal directement sur la sortie audio, les deux suivantes règlent le volume et la hauteur du signal généré, et la dernière demande d'attendre deux secondes avant d'arreter.

Mais je suis bien d'accord avec vous, ce son est un peu agressif, surtout dans l'attaque. Hop, nous allons brancher un nouveau module entre le générateur et la sortie : un contrôleur d'enveloppe qui atténuera la brutalité de l'attaque et la violence de la fin du son.

```
// Just a better beep
SinOsc s  => Envelope e   =>   dac;
// reglage de notre oscillateur
0.5         =>   s.gain;
440         =>   s.freq;
// reglage de notre enveloppe
200::ms => e.duration;
```

```
// hop !
e.keyOn();
2::second   =>   now;
e.keyOff();
1::second   =>   now;
```

C'est déja mieux *:)* Détaillons les quatre dernières ligne de ce script : deux nous sont déja connues, celles où l'on met une *duration* dans le `now` afin d'attendre un certain temps. quand aux deux autres (qui sont des appels de méthode, reconnaissables à la paire de parenthèses qui les terminent) elles déclenchent le debut et la fin du contrôleur d'enveloppe.

Quand on actionne le `keyOn`, le processeur d'enveloppe commence avc un gain à 0 et monte progressivement au gain maximal dans le temps réglé au préalable dans `e.duration`. De la même façon, `keyOff` fait redescendre le volume à 0 à la fin du beep sur la même durée, et pour entendre cette partie finale , il nous faut attendre un peu, d'où la dernière ligne.

Et avant de passer à la suite, voyons un peu le coté plus « informatique » de Chuck. La séquence d'instruction finale de l'exemple précédent peut être vu comme un bloc unique, puisque il n'a qu'une seule finalité : générer le beep.

Et pour ça, nous allons les regrouper dans une **fonction** que voici que voilà :

```
fun void beep() {
e.keyOn();          2::second   =>   now;
e.keyOff();         1::second   =>   now;
}
```

Pour l'utilisation de cette fonction, c'est dans la partie suivante. Bienvenue dans le monde merveilleux des lignes de code.

7.1.1 Les filtres

Le son que nous avons utilisé dans les précédents exemples est une sinusoïde, un son « pur » que l'on peut qualifier sans crainte de légèrement pauvre. Nous allons régler ça.

Vous pouvez déja essayer de remplacer le sinus par une onde carrée, un `SqrOsc`, bien remplie d'harmoniques. C'est ce que nous allons faire, mais nous allons aussi rajouter un filtre passe-bande pour moduler la couleur du son.

```
//--- Just a noisy beep
SqrOsc s => BPF f => Envelope e => dac;
//--- reglage de notre oscillateur
0.8         =>   s.gain;
440         =>   s.freq;
//--- reglage de notre enveloppe
100::ms     =>   e.duration;
//--- reglage du filtre passe bande
2           =>   f.Q;
1337        =>   f.freq;
```

```
fun void beep() {
e.keyOn();          2::second   =>   now;
e.keyOff();         1::second   =>   now;
}

           () => beep;
```

Puisque la fréquence de notre *beep* est calée à 440 Hz, nous allonc chercher les harmoniques supérieures vers 1337 Hz, avec une largeur de bande assez grande. Et nous obtenons un son bien plus brillant, mais encore un peu inerte.

Prochaine étape, brancher un LFO sur ce filtre...

7.1.2 Beep à vibrato

Les choses vont un peu se corser, avec la notion de *multi thread*. Un grand mot pour une chose (relativement) simple. En fait, une partie du script va s'exécuter indépendamment du reste, un peu comme l'autonomie de votre LFO matériel.

```
SqrOsc s => BPF f => Envelope e => dac;
0.8         =>   s.gain;
440         =>   s.freq;
100::ms     =>   e.duration;

//--- reglage du filtre passe bande
2           =>   f.Q;
1337        =>   f.freq;

          () => beep;
```

Et nous retrouvons la notion de fonction.

7.1.3 Attack Decay Sustain Release

Un module d'enveloppe bien plus élaboré, fidèle aux canons de l'époque. Son utilisation est semblable à `Envelope` que nous avons vu plus haut, ce qui nous donne ce patch :

```
SqrOsc s => BPF f => ADSR adsr => dac;
```

Point suivant : configurer notre ADSR. Il y a bien plus de potards que dans l'exemple précédent : attack time et rate, decay time et rate, release time et rate et sustain level pour finir.

7.1.4 Jouer des samples

Oui, on peut. On peut même faire des choses assez fun. Juste un example un peu brouillon et qui manque de rigueur, essentiellement sur le calcul de la durée d'écoute de l'échantillon en fonction du *pitch*. Mais, en attendant mieux :

```
SndBuf buf => dac;
"/home/tth/Essais/Chuck/sample.wav" => buf.read;

float   srate, duration;
1 => buf.gain;

for ( 0.7=>srate; srate<1.9; 1.1 *=> srate) {
        4000 / srate => duration;
        <<< "loop...", srate, duration >>>;
        0 => buf.pos;
        srate => buf.rate;
        (duration)::ms => now;
}
```

Bien entendu, il vous faudra créer le fichier avec l'échantillon sonore. J'ai utilisé un fichier de quatre secondes échantilloné à 22050 Hz.

Et j'entend un esprit chagrin râler au fond de la salle : « oué, nous on veut du gros son ». Patience, nous allons d'abord voir les briques avant de construire le mur.

7.1.5 Dialoguer en MIDI

Oui, on peut. Et nous allons faire des beep à partir d'un clavier ou du séquenceur d'une electribe. Attention, une petite connaissance du protocole MIDI est nécessaire. Vous trouverez quelques explications page 43.

Le petit bout de code qui suit vous affiche les triplets reçus en charabia numérique. Leur décriptage est laissé en exercice à nos lecteurs.

```
MidiIn          input;
MidiMsg         msg;

1 => int port;                  // please explain

if (!input.open(port)) {
        <<< "erreur open MIDI" >>>;    me.exit();
        }

while (1) {
        input => now;
        while ( input.recv(msg) ) {
                <<< msg.data1, msg.data2, msg.data3 >>>;
                }
        }
```

Cet exemple est volontairement simple, voire même minimaliste, afin de ne pas surcharger cet ouvrage. Mais je vous rappelle que les exemples sur le site sont très instructifs.

7.1.6 Communication OSC

OSC, Open Sound Control, a son chapitre dédié en page 48. Nous allons nous concentrer sur les aspects Chuck. J'ai bien une idée d'exemple, mais ce soir, c'est soirée "flemme".

7.1.7 Enregistrer

Il est aussi possible de générer des fichiers .WAV à partir d'un script Chuck.

7.1.8 Aller plus loin

Et pour la suite, allez onsulter http ://chuck.cs.princeton.edu/ et http ://-chuck.stanford.edu/ pour tout savoir. La rubrique des exemples est très instructive.

7.2 Super Collider

Là, nous rentrons dans le domaine de la jeune Nina, je vais donc lui,laisser la plume (ou le clavier).

7.3 CSound

Chapitre 8

Musical Instrument Digital Interface

Dans un premier temps, je vais me concentrer sur les services MIDI offerts par la couche ALSA d'un système Linux. Ensuite, nous passerons aux *raw devices* (page 45). Mais avant tout, un petit rappel sur ce qu'est le MIDI.

> *Wikipedia* : C'est un protocole de communication et un format de fichier dédiés à la musique, et utilisés pour la communication entre instruments électroniques, contrôleurs, séquenceurs, et logiciels de musique. Apparu dans les années 1980, issu d'une volonté de l'industrie de normaliser les échanges entre les différents instruments, ce protocole est devenu aujourd'hui un standard très répandu dans le matériel électronique de musique.

Attention, ce système **ne** transporte **pas** de signal audio, mais de petits messages (quelques octets, en général 3) de **commande** informatique [1]. Un exemple de message basique, c'est celui-ci, exprimé en hexadécimal : `93 5B 3A` un message qui va déclencher, sur le canal 3, la note Sol5, à un niveau assez bas. Et c'est l'instrument à l'autre bout du fil qui va générer le signal audio.

8.1 Seq24

Seq24 has a unique convention for entering and editing notes that seems confusing at first but quickly makes sense once you use it once or twice.

- Right-click to activate the pen tool, and left-click (while still holding the right-click button) to draw a note.
- Left-click to select a note. Left-click and drag to draw a selection box.
- Middle-click and drag to modify the length of a selected note.

1. C'est une confusion fréquente qui complique les choses

8.2 Outils de base

Le sous-système ALSA vient avec quelques utiltaires bien pratique dans un usage d'exploration.

8.2.1 aseqdump

Cet outil permet de voir de façon relativement claire **tous** les évènements MIDI qui rentre par un chemin donné, et, en première étape, surtout de lister ces chemins.

```
tth@plop:~/Muzak/Samples$ aseqdump -l
 Port    Client name                      Port name
  0:0    System                           Timer
  0:1    System                           Announce
 14:0    Midi Through                     Midi Through Port-0
 20:0    nanoKONTROL                      nanoKONTROL MIDI 1
```

Voici ce qu'on obtient quand on remue les boutons d'un nanoKontrol de Korg, et voyez bien qu'on retrouve les trois octets dont on causait l'autre jour. Le premier contient à la fois le numéro ce canal et le type d'évènement, le second le numéro du contrôle concerné, et le troisième la nouvelle valeur :

```
tth@fubar:~/Documents/TetaTricks$ aseqdump -p 20
Waiting for data. Press Ctrl+C to end.
Source  Event                  Ch  Data
 20:0   Control change          1, controller 45, value 127
 20:0   Control change          1, controller 45, value 0
 20:0   Control change          1, controller 14, value 120
 20:0   Control change          1, controller 14, value 119
```

Un exemple d'utilisation d'un nanoKontrol pour piloter Chuck se trouve (peut-être) aux alentours de la page 41.

8.2.2 aconnect

Maintenant que nous savons ce qui rentre dans notre machine, il est temps de savoir comment le faire ressortir. Il faut donc créer une connexion quelque part au tréfond des tables obscures de la kernelle.

8.2.3 aseqnet

> aseqnet is an ALSA sequencer client which sends and receives event packets over the network.

8.2.4 pmidi

Jouer un fichier MIDI (aka *midifile*) au travers du séquenceur ALSA.

8.3 ALSA tips

cat /proc/asound/seq/clients

```
Client info
  cur  clients : 4
  peak clients : 4
  max  clients : 192

Client   0 : "System" [Kernel]
  Port   0 : "Timer" (Rwe-)
  Port   1 : "Announce" (R-e-)
    Connecting To: 15:0
Client  14 : "Midi Through" [Kernel]
  Port   0 : "Midi Through Port-0" (RWe-)
Client  15 : "OSS sequencer" [Kernel]
  Port   0 : "Receiver" (-we-)
    Connected From: 0:1
Client 128 : "TiMidity" [User]
  Port   0 : "TiMidity port 0" (-We-)
  Port   1 : "TiMidity port 1" (-We-)
  Port   2 : "TiMidity port 2" (-We-)
  Port   3 : "TiMidity port 3" (-We-)
  Output pool :
    Pool size          : 500
    Cells in use       : 0
    Peak cells in use  : 0
    Alloc success      : 0
    Alloc failures     : 0
  Input pool :
    Pool size          : 1000
    Cells in use       : 0
    Peak cells in use  : 0
    Alloc success      : 0
    Alloc failures     : 0
```

8.4 Raw devices

Plutôt que de passer par la couche ALSA, il est aussi possible d'attaquer directement le *device*, et de blaster soi-même tous les petits octets directement sur le fil. Voici un petit exemple en Python [2], facilement transposable dans un langage plus sérieux :

```
#!/usr/bin/env python

import time

def note_on(note):
        tx.write("%c%c%c" % (0x90, note, 100))
```

2. Oui, je sais, c'est un langage de djeunz, mais bon, ça fait plaisir à mon gamin

```
        tx.flush()
def note_off(note):
        tx.write("%c%c%c" % (0x90, note, 0))
        tx.flush()
def progchange(prog):
        tx.write("%c%c" % (0xc0, prog))
        tx.flush()

tx = open("/dev/midi", "w")
for valeur in range(30,90):
        progchange(valeur)
        note_on(valeur)
        time.sleep(1)
        note_off(valeur)
        time.sleep(1)

tx.close()
```

8.5 Et en Perl ?

En Perl[3], il existe un certain nombre de modules très évolués pour jouer avec les fichiers MIDI, ce qui est un domaine légèrement différent, il faut le reconnaitre.

```
MIDI (3pm)           - read, compose, modify, and write MIDI files
MIDI::Event (3pm)    - MIDI events
MIDI::Filespec (3pm) - MIDI File Specification
MIDI::Opus (3pm)     - functions and methods for MIDI opuses
MIDI::Score (3pm)    - MIDI scores
MIDI::Simple (3pm)   - procedural/OOP interface for MIDI composition
MIDI::Track (3pm)    - functions and methods for MIDI tracks
```

Un petit essai, par exemple extraire la valeur du BPM d'un fichier `.mid` ?

```
#!/usr/bin/perl -w
use MIDI;
0;
```

Voilà enfin du code qui n'est pas fini...

8.6 ABC notation

ABC, développé par Chris Walshaw, est un format conçu pour noter la musique en utilisant du texte brut. Il a été conçu à l'origine pour les airs folkloriques d'origine ouest-européenne qui peuvent être écrits sur une seule portée, mais a depuis été étendu à la notation de partitions complètes de diverses musiques.

Un exemple :

3. Oui, je sais, c'est un langage de vieux, mais bon, il faut bien se faire plaisir de temps en temps...

```
X:42
T:Paddy O'Rafferty
C:Trad.
M:6/8
K:D
dff cee|def gfe|dff cee|dfe dBA|dff cee|def gfe|faf gfe|1 dfe dBA:|2 dfe dcB||
~A3 B3|gfe fdB|AFA B2c|dfe dcB|~A3 ~B3|efe efg|faf gfe|1 dfe dcB:|2 dfe dBA||
fAA eAA|def gfe|fAA eAA|dfe dBA|fAA eAA|def gfe|faf gfe|dfe dBA:|
```

Je vous préviens d'entrée : je découvre l'univers ABC en même temps que j'écris ces lignes. Je vais donc raconter plein de bétises [4] du genre Ne faites pas ça à la maison.

8.6.1 Header

8.6.2 Partition

8.7 midish

Vous avez aimé *packet filter* ? Alors vous allez adorer midish...

8.8 Et après ?

OSC (page 48) semble une possibilité pour aller au-delà. Mais, *amho*, rien ne pourra vraiment remplacer le midi tel que nous le connaissons depuis les années 80.

You can read more about USB MIDI here : https ://www.midi.org/midi-articles/basic-of-usb

4. Ça va, on est au courant.

Chapitre 9

Open Sound Control

Wikipedia : *OSC est un format de transmission de données entre ordinateurs, synthétiseurs, robots ou tout autre matériel ou logiciel compatible, conçu pour le contrôle en temps réel. Il utilise le réseau IP au travers des protocoles UDP ou TCP et apporte des améliorations en termes de rapidité et de flexibilité par rapport à l'ancienne norme MIDI...*

Actuellement, j'utilise (pour le C) `liblo` qui reste simple à utiliser tout en permettant de descendre à un bas niveau. Cette bibliothèque est livrée avec deux outils que nous allons voir maintenant.

9.1 oscdump & oscsend

Envoyer :

Pour envoyer, il faut bien définir une destination. Une fois de plus, le couple IP adresse/port est utilisé. Ce sont les deux premiers paramètres.

```
tth@debian:~$ oscsend localhost 1664 /tmp/kmem is 61 'meg, efface !'
tth@debian:~$
```

La chaine `is` que vous voyez dans la ligne de commande décrit le type des paramètres qui suivent : `i` est pour un integer et `s` pour une chaine (notez les simples quotes pour passer les caractères speciaux du shell).

Ecouter :

```
tth@debian:~$ oscdump 1664
/tmp/kmem is 61 "meg, efface !"
^C
tth@debian:~$
```

9.2 Exemple de code

Ecouter :

Attention, là, nous arrivons sur un domaine plus hardu [1]. En effet, la mécanique interne de `liblo` utilise un thread d'écoute qui appelle des *callbacks* préalablement "accrochés", *hooked* en vrai langue.

```
/*      LISTEN OSC                                      */

#include  <stdio.h>
#include  <lo/lo.h>

#define LOCAL_PORT      "9000"

int main(int argc, char *argv[])
{
lo_server_thread st;

st = lo_server_thread_new(LOCAL_PORT, NULL);

return 0;
}
```

Envoyer :

Cette partie là est beaucoup plus simple.

```
/*      SEND OSC                                        */

#include  <stdio.h>
#include  <lo/lo.h>

#define REMOTE_HOST     "localhost"
#define REMOTE_PORT     "9000"

int main(int argc, char *argv[])
{
lo_address      loana;
int             foo;

loana = lo_address_new(REMOTE_HOST, REMOTE_PORT);
foo = lo_send(loana, "/dev/kmem", "is", 61, "meg, efface !");
fprintf(stderr, "foo %d\n", foo);
return 0;
}
```

J'avoue ne pas encore savoir ce que veut bien dire la valeur de retour.

9.3 En Perl

```
use Net::OpenSoundControl::Client;

$client =  Net::OpenSoundControl::Client->new(
                  Host => $target, Port => $port)
or die "Could not start client: $@\n";
```

1. Un beau néoligisme, n'est-il pas ?

```
$client->send(['/proc/kcore', 'i', 61, 's', "bla"]);
```

9.4 Le routage

WIP...

9.5 Autres logiciels

Beaucoup d'autres logiciels peuvent être controlés par OSC, ou générer des trames OSC : on peut citer Processing, Chuck (page 42), ...

Chapitre 10

Image

Le traitement des images est un art à part entière. Nous allons voir quelques grands classiques de la catégorie pas du tout interactif. Le genre de machin que l'on peut ranger au fond d'un script shell pour le réutiliser la semaine prochaine.

- ImageMagick
- Gmic
- NetPBM
- Portable Network Graphic
- TIFF

10.1 ImageMagick

Attention, ça va devenir *hardu*[1], l'abondance d'options des outils ImageMagick est vraiment énorme, et leurs interactions parfois troublantes...

Il y a plusieurs binaires dans le paquet : `convert` pour la plupart des traitements, `display` et `animate` pour visualiser des images, `identify` pour en lire les méta-données.

Ils partagent un certain nombre d'options communes. Certains d'entre eux peuvent avoir un effet destructif[2], vous êtes prévenus.

10.1.1 L'option `-geometry`

C'est une notion importante, car elle est utilisée comme paramètre par beaucoup d'options. C'est avec ça que l'on peut définir des choses comme la taille (largeur et hauteur) d'une image, ou la position d'un texte dans l'image, ou une zone sur laquelle on va travailler.

Et c'est assez compliqué, mais pas que.

1. Nous attendons tous avec impatience l'ouvrage magistral de maitre Brunus
2. En plus d'avoir un nom de série *Z*.

Code	Signification
scale%	Height and width both scaled by specified percentage.
scale-x%xscale-y%	Height and width individually scaled by specified percentages. (Only one % symbol needed.)
width	Width given, height automagically selected to preserve aspect ratio.
xheight	Height given, width automagically selected to preserve aspect ratio.
widthxheight	Maximum values of height and width given, aspect ratio preserved.
widthxheight^	Minimum values of width and height given, aspect ratio preserved.
widthxheight!	Width and height emphatically given, original aspect ratio ignored.
widthxheight>	Shrinks an image with dimension(s) larger than the corresponding width and/or height argument(s).
widthxheight<	Enlarges an image with dimension(s) smaller than the corresponding width and/or height argument(s).
area@	Resize image to have specified area in pixels. Aspect ratio is preserved.
x :y	Here x and y denotes an aspect ratio (e.g. 3 :2 = 1.5).

10.1.2 Écrire du texte

Voici un exemple concret (tiré du script d'encodage du CLOÎTRE) qui montre l'essentiel pour bien débuter. C'est une fonction écrite en bash qui rajoute un texte sur une image :

```
#!/bin/bash

function tagpic
{
infile="$1"
outfile="$2"
texte="$3"
txtfont="␣-font␣Utopia␣"
fontsize="␣-pointsize␣96␣-kerning␣4␣"
colors="␣-fill␣Gray20␣-stroke␣White␣"
txtopts="␣-antialias␣-alpha␣off␣$txtfont␣"

convert          $infile                              \
                 ${txtopts}                           \
                 ${txtfont} ${fontsize}               \
                 ${colors}                             \
                 -gravity South                       \
                 -annotate +0+85 "${texte}"           \
                 $outfile
```

```
}

if [ "$#" -eq 1 ] ; then
        tagpic foo.png bar.png "$1"
        display bar.png
fi
```

Une petite commande toute bête pour rajouter la date facilement à une photo d'après les données EXIF :

```
convert nom_du_fichier_source.jpg -gravity SouthEast        \
        -font Fudd-Regular -pointsize 40 -fill white        \
        -annotate +20+20 "%[exif:DateTimeOriginal]"         \
        -quality 100 nom_du_fichier_destination.jpg
```

Quelques explications semblent nécessaires, parce que certaines options sont un peu ésotériques et parfois inter-dépendantes...

Code	Signification
-txtfont	
-fontsize	taille des caractères
-kerning	espacement entre les caractères
-color	couleur de l'encre
-stroke	couleur 'bord' de caractère
-strokewidth	largeur de la bordure
-txtopts	*Gni ?*
-gravity	vers quel point cardinal le texte va-t-il être *tiré* ?
-annotate	`+30+30 "texte à écrire"`

Pour avoir la liste des couleurs disponibles : `convert -list color` et celles des polices de caractère : `convert -list font`.

- Comment mettre une couleur de fond à ce texte ?
- Et un texte semi-transparent ?

10.1.3 montage

> create a composite image by combining several separate images. The images are tiled on the composite image optionally adorned with a border, frame, image name, and more.

Krkrkr...

10.1.4 Faire des GIFs animées

Ces petites images clignotantes sont l'essence même du Web moderne depuis 1992 malgré leurs limitations, et `convert` sait très bien [3] les générer. Encore un grand merci à Compuserve pour cette création essentielle à l'art graphique moderne.

3. modulo un taux de compression pas génial.

```
convert -delay 20 -loop 0 a.png b.png c.png foo.gif
```

Ce sujet pertinent est abordé plus en détails en page 141 avec, nous l'espérons, un exemple de dithering avec la palette de couleur `EGA`.

10.1.5 Extraire une partie de l'image

Dans cette fonctionnalité aussi, il semble y avoir quelques subtilités.

```
DIMENSION="1024x768"
POSITION="+R12+0"
convert -crop ${DIMENSION}${POSITION} +repage $img $dst
```

Use +repage to completely remove/reset the virtual canvas meta-data from the images. En gros, on en a besoin quand une opération va changer la dimension de l'image. En détail, j'ai pas tout compris.

10.1.6 conversion vers TGA

On trouve encore en circulation de très vieux logiciels qui ne savent lire qu'un encore plus ancien format : *Targa file format*. Et dans une variante bien précise : mode RGB et non compressé [4].

Il nous faut donc une incantation bien précise avec, heureusement, des arguments aux noms évocateurs :

```
GOTO_TGA ()
{
convert $1                              \
        -compress none                  \
        -alpha off                      \
        -colorspace RGB                 \
        -type truecolor                 \
        $2
}
```

On peut aussi procéder autrement, avec les outils PNM :

```
pngtopnm \$negatif | ppmtotga -rgb -norle > \$picz
```

10.1.7 identify

Cette commande nous fournit un ensemble d'informations [5] pertinentes sur un fichier image : format, résolution, couleur ou n/b, nombre de bits par pixel...

```
tth@konrad:~/GravityField $ identify WS/nanim//00000.pgm graph.png
WS/nanim//00000.pgm PGM 2048x2048 2048x2048+0+0 16-bit Grayscale
                                        Gray 20.5765MiB 0.690u 0:00.690
graph.png PNG 640x640 640x640+0+0 8-bit sRGB 102c 18165B 0.000u 0:00.000
```

4. Un jour, peut-être, ça va changer...
5. aka metadata

https ://imagemagick.org/script/escape.php

Et ces données peuvent être utilisé pour rajouter une simili IA [6]

10.1.8 Configuration

La configuration est dans le répertoire `/etc/ImageMagick-6/`, et on y trouve en particulier le fichier `policy.xml` [7] où certaines ressources, comme la mémoire utilisable, peuvent être réglées.

```
$ identify -list resource
Resource limits:
  Width: 16KP
  Height: 16KP
  List length: 18.446744EP
  Area: 128MP
  Memory: 256MiB
  Map: 512MiB
  Disk: 1GiB
  File: 768
  Thread: 4
  Throttle: 0
  Time: unlimited
```

Vous pouvez aussi tenter de comprendre le contenu du fichier `thresholds.xml` puis revenir me l'expliquer juste après ma sieste, je pense que ça permet de jouer sur différentes méthodes de *dithering*, par exemple pour faire du Bayer.

10.1.9 Trucs

- convert -average frame*.png output.png
- convert -brightness-contrast 20% foo.png bar.png

Il y en plein d'autres, des tricks *àlc* à découvrir. Ce sera, pour vous amis lecteurs, une dure mission à assumer.

10.1.10 Ressources

Le futur livre de Brunus.

Réclamez les premiers drafts dans l'IRC `libera.chat#paulla` *:)*

`http://www.fmwconcepts.com/imagemagick/index.php`

10.2 Gmic

Perform generic image processing operations, through the G'MIC language interpreter. gmic : GREYC's Magic for Image Computing.

6. Sorry, private joke inside...
7. Le XML, c'est comme la violence...

Gmic se base sur une grosse [8] bibliothèque de fonctions écrite en `C++` dont l'utilisation est bien documentée dans le livre ? ? ? mais nécessite quand même de solides bases dans les subtilitées fourbes du `C++`.

Gmic existe aussi en plugin pour Gimp avec une interface pleine de curseurs et de boutons, qui facilite (ou pas) l'exploration des possibles.

10.3 NetPBM

With the Netpbm file formats, it's trivial to output pixels using nothing but text based IO [9].

```
#!/bin/bash
exec > my_image.ppm    # All echo statements will write here
echo "P3 250 250 255"  # magic, width, height, max component value
for ((y=0; y<250; y++)) {
  for ((x=0; x<250; x++)) {
    echo "$((x^y)) $((x^y)) $((x|y))" # r, g, b
  }
}
```

10.3.1 Questions

Comment faire un *resize* avec NetPBM ? Est-il nécessaire d'apprendre la bibliothèque de fonctions `libnetpbm` [10] ?

10.4 Portable Network Graphic

Voici le problème du soir [11] : j'utilise pour lire ce format de fichier la bibliothèque `pnglite` qui a de gros soucis avec beaucoup de ceux-ci, par exemple ceux qui sont en `sRGB` crées par des outils essentiels comme POVray. Il serait bon de voir la `libpng` officielle.

pngcrush ? optipng ?

```
pngtopnm $png | pnmtopng > $goodpng
```

10.5 Tag Image File Format

> Yes, but tiffcp is by far the most significant case. And perhaps one of the most used of the tools. I may not be impartial on this, though :-) because I use it heavily.

```
tiffcp  combines  one  or more files created according to the Tag Image
File Format, Revision 6.0 into a single TIFF file.  Because the  output
```

8. énorme, même. dans les deux sens. . .
9. https ://www.vidarholen.net/contents/blog/ ?p=904
10. Réponse : oui.
11. *trollday* Fri Nov 13 00 :07 :19 CET 2020

```
file may be compressed using a different algorithm than the input
files, tiffcp is most often used to convert between different compres-
sion schemes.
```

10.6 EXIF

Un *tag* EXIF est un petit morceau d'information nommée inclus dans un fichier image. Il peut contenir des données comme la sensibilité ISO du capteur, ou le modèle d'objectif utilisé.

D'autres informations en page 129

10.7 Cimg

`Cimg` est l'énorme bibliothèque de fonctions (écrites en `C++` [12]) sur lesquelles est basé Gmic. C'est long à compiler, ça produit des binaires énormes, mais il semble bien que ça puisse faire des trucs géniaux.

Peut-être [13], existe-il une interface pour le Fortran moderne ?

10.7.1 HELLO WORLD

```
#define cimg_use_png
#include "CImg.h"
using namespace cimg_library;

#define BLUR 0.35
int main(int argc,char **argv)
{
if (3 != argc) {
        fprintf(stderr,                                \
            "Usage: %s infile.png outfile.png\n", argv[0]);
        exit(1);
        }
CImg<unsigned char> imgIn (argv[1]);
CImg<unsigned char> imgOut =                           \
             imgIn.get_norm().blur(BLUR).normalize(0, 255);
imgOut.save(argv[2]);
return 0;
}
```

Un concept interessant, mais pas évident au premier abord. Un peu comme les branchements dans Chuck ou les pipes du shell...

12. WTF ?
13. Les rêves ne sont pas interdits.

10.7.2 Questions...

- Comment virer le canal alpha à l'enregistrement d'une image en PNG ?

Chapitre 11

Les Intertubes

Ah, le grand Internet sauvage, il serait temps qu'on en parle un peu. Nous allons voir dans ce chapitre quelques utilisations que l'on peut qualifier de « créatives ».

Si vous attendiez des choses plus sérieuses, j'ai aussi ça en magasin : DNS page 146, SSH page 148, streaming page 65, et probablement bien d'autres à venir.

11.1 xinetd

Xinetd est un *super-server daemon* qui gère l'accueil et le lancement pour certains services Internet essentiels tels que `daytime` ou `echo`. Par rapport à son ancêtre `inetd`, décrit en page 111, il offre plus de souplesse dans la configuration et la gestion des droits d'accès.

Ceci dit, de nos jours [1], il a probablement été remplacé par quelque chose d'encore plus étrange basé sur *systemd*, puisque l'important ça n'est pas que ça soit mieux, mais que ce soit **nouveau**.

11.1.1 qotd

À titre d'exercice, nous allons reprendre l'exemple du serveur de *quote of the day* que nous avons déja utilisé avec Inetd, et lancer le petit script par l'intermédiaire de Xinetd.

Pour cela, il faut créer dans le répertoire `/etc/xinetd.d/` [2] un fichier nommé `qotd` et contenant ceci :

```
service qotd
{
disable         = yes
type            = UNLISTED
```

1. Wed Nov 11 04 :47 :50 CET 2020
2. Attention, ce répertoire est peut-être un debianisme, ymmv.

```
socket_type     = stream
protocol        = tcp

user            = nobody
server          = /usr/local/bin/qotd.sh
}
```

Bien entendu, le script `qotd.sh` décrit en page 111 sera placé dans le bon répertoire. Ensuite, il faut demander à Xinetd de relire ses fichiers de configuration par un signal bien senti et au bon endroit.

Ah, on me murmure dans l'oreille que Xinetd n'existe plus dans les Linux modernes, et qu'il existe quelque chose de *beaucoup mieux*. Je ne sais pas quoi. Je demande à voir [3].

11.2 Translation d'adresse

Voir en page 168 pour découvrir comment utiliser une machine connectée en WiFi pour donner accès à Internet à votre réseau local Ethernet...

Mais cette translation d'adresse peut avoir aussi d'autres défis à relever pour en faire une solution complexe et amusante à des problèmes inexistants et datant du siècle dernier. Par exemple la comparaison de la mise en œuvre du NAT avec `iptables` et `pfctl` est une source de trolls sans fin.

11.3 Iodine

Iodine, c'est quoi ? C'est un ensemble de logiciels qui permettent de se brancher sur l'Intertube même si on n'a pas d'accès « autorisé » par la voisine qui a déménagée vers la lointaine campagne avec sa friboite.

> This is a piece of software that lets you tunnel IPv4 data through a DNS server. This can be usable in different situations where internet access is firewalled, but DNS queries are allowed. The name iodine was chosen since it starts with IOD (IP Over DNS) and since iodine has atomic number 53, which happens to be the DNS port number.

À ce jour (2 septembre 2013) ce que vous allez lire est basé sur la version `0.6.0-rc1` et sera probablement plus ou moins incompatible avec les versions précedentes ou suivantes, ymmv.

```
root@plop:iodine-0.6.0-rc1# bin/iodine teton.mooo.com -f -P m
Opened dns0
Opened UDP socket
Sending DNS queries for teton.mooo.com to 192.168.1.1
Autodetecting DNS query type (use -T to override).................
iodine: No suitable DNS query type found. Are you connected to a network?
iodine: If you expect very long roundtrip delays, use -T explicitly.
iodine: (Also, connecting to an "ancient" version of iodined won't work.)
```

3. Parce que c'est écrit par LP ?

11.3.1 Coté serveur

Je pense qu'il faut commencer par voir du coté du DNS pour déléguer une sous-zone à la machine qui va faire tourner le serveur iodine. Mais j'en suis pas certain. RTFM.

11.3.2 Coté client

Je n'en sais pas plus non plus...

```
sudo bin/iodine -f -P s3cr3tp4ssw0rd i.buvette.org
```

Donc, n'ayant plus vraiment (pour le moment) le besoin d'utiliser ce passe-muraille, je vais laisser ce passage en l'état...

11.4 Outils de diagnostic

iftop, iptraf ...

11.4.1 iperf

Avec `iperf`, on peut mesurer le débit du réseau IP (en TCP ou UDP) entre deux machines. Il faut d'abord le lancer sur la machine cible en mode "serveur" ou "daemon". Ensuite, des clients peuvent mesurer le débit entre eux-même et le serveur.

Le port par défaut utilisé est le 5001.

11.4.2 tcpdump

C'est clairement un outil dédié au voyeurisme, mais qui permet de décrypter les ésotérismes de MiniMad. C'est donc sur cet exemple que nous allons partir.

Ces gadgets dialoguent entre eux par le protocole *Bonjour*, aka *ZeroConf*[4].

Mais en fait, non. Je n'ai plus les Minimad sous la main. Il va falloir trouver un autre exemple, ou l'imaginer nous-même.

11.4.3 nmap

C'est clairement aussi un outil dédié au voyeurisme à distance, puisque il permet de découvrir, avec plus ou moins de détails, les services exposés à l'extérieur par une machine distante.

Comment scanner un /24 pour y découvrir les machines ?

```
# nmap -sS 192.168.42.0/24
```

`-sS` = TCP SYN scan.

4. C'est en page 133 qu'on en parle vaguement

11.5 netcat

Attention : Il existe au moins N variantes de netcat en circulation, bien entendu avec des options et des comportements différents.

> The nc (or netcat) utility is used for just about anything under the sun involving TCP or UDP. It can open TCP connections, send UDP packets, listen on arbitrary TCP and UDP ports, do port scanning, and deal with both IPv4 and IPv6 [5]. Unlike telnet(1), nc scripts nicely, and separates error messages onto standard error instead of sending them to standard output, as telnet(1) does with some.

Netcat was written by a guy we know as the Hobbit <hobbit@avian.org>.

5. Voir le chapitre **??** pour cette nouveauté.

Chapitre 12

Perl

Également connu sous le nom de *Pathologically Eclectic Rubbish Lister*[1], Perl est un langage qui a été crée en 1987 par Larry Wall comme un substitut moderne au classique combo shell/sed/awk. Il est de ce fait très orienté vers la manipulation de données textuelles et contient d'énormes facilités d'utilisation des expressions régulières.

Il fut un temps, dans le siècle précédent, où il était le maitre incontesté du Web interactif, de par son intégration dans le serveur Apache, et de modules d'écriture de clients http.

Le débutant peut facilement n'utiliser qu'un sous-ensemble du langage, mais ce serait se priver de quelques *patterns* assez efficients.

12.1 Le texte

Un cas concret : générer du code en langage `C` à partir d'un fichier traité ligne par ligne. Chacune de ces lignes est de la forme `ID un texte quelconque.` avec ID qui est une courte chaine alphabétique. Le résultat de la conversion sera `char *p_ID "un texte quelconque.";` et pourra être utilisé par un `#include "fichier.h"` bien classique.

12.1.1 Le traitement

Ce traitement peut être décomposé en deux actions emboitées : d'une part traiter une ligne, et d'autre part, traiter un fichier plein de lignes.

La première va être encapsulée dans un module Perl.

```
#!/usr/bin/perl -w
use strict;
sub TraiterLigne($)
{
my $in = shift;
print "input: $in\n";
```

1. collectionneur pathologique de déchets variés

```
}
1;
```

12.1.2 L'interface

Quand à l'interface utilisateur, quelques arguments sur la ligne de commande feront l'affaire.

```
#!/usr/bin/perl -w
use strict;
use code::id2h;

TraiterLigne("ID␣un␣texte␣quelconque.");
```

12.2 Gestion des modules

Tout comme pour le Fortan Moderne, c'est une notion essentielle.

12.3 Le `http`

Scrapping de la patate en vue !

https ://al.smeuh.org/cgi-bin/express.pritz

12.4 Et en vrai ?

`exiftool - Read and write meta information in files`

12.5 RFC 8589

RFC 8589 is an "informational" RFC that serves as a guide for developers of applications, such as email clients, Web browsers and more, to implement the "leaptofrogans" URI scheme (see also the official URI schemes registered with the IANA). Nobody is under any obligation, but RFC 8589 establishes a clear path for enabling direct links to Frogans sites in online content.

12.6 oneliners

Chapitre 13

streaming

Avertissement : pour le moment [1], nous n'allons nous pencher que sur la diffusion de flux audios, avec des encodages libres comme *ogg* ou *speex*. Tout simplement parce que le monde de la vidéo est un gigantesque foutoir de codecs, de containers, de brevets logiciels et d'incompatiblités diverses et variées.

Une notion importante à bien comprendre, c'est l'organisation du flux des sons immatériels au travers des intertubes.

insert picture here !

13.1 icecast2

Icecast est, semble-t-il, **le** serveur de streaming de référence. Il se configure facilement avec des fichiers en XML. Première approche d'une ergonomie différente, vous êtes avertis.

> Icecast, the project, is a collection of programs and libraries for streaming audio over the Internet. This includes : Icecast, the program that streams audio data to listeners, libshout, a library for communicating with Icecast servers and Ices2, a program that sends audio data to Icecast servers

Pour alimenter Icecast en son, il faut utiliser un *source-client* approprié. Le modèle recommandé est **ices2** qui est détaillé un peu plus loin, à la page 68.

Il peut envoyer depuis une entrée audio (alsa, jack, pulseaudio...) ou une playlistde fichiers `.ogg`.

more to come...

13.2 liquidsoap

Mais nous allonc commencer par un outil puissant, bien que délicat à comprendre au premier abord : `https://www.liquidsoap.info/`

1. En juillet 2014

> Liquidsoap is an extensible and flexible audio stream generator, mainly used for streaming to a shoutcast/icecast server. It is also possible to output in a file, to speakers or as raw PCM stream.
>
> liquidsoap can perform any audio processing, mixing streams together, applying filters, generating sound procedurally, there is no limit but your imagination. Input files can be accessed through the network, using samba, ftp or speech synthesis. The system features lots of programming operators, including user requests, time-based selection, sum...

Un étonnant langage de script pour gérer tous ces octets musicaux en les promenant un peu dans tous les sens, la possibilité de commander Savonet avec un simple telnet. Vite, un exemple d'utilisation, vous créez un fichier texte avec cez quelques lignes dedans et vous le sauvez sous le nom `savon.liq` :

```
set("log.file.path","/tmp/savon.log")
set("log.stdout",true)

output.icecast (%vorbis,
                host = "10.20.0.23",
                port = 1984,
                password = "hopla!",
                mount = "savon.ogg",
                single("loveme.ogg"))
```

Avec ce petit programme, nous pouvons envoyer en boucle un fichier Ogg vers notre serveur. Les deux premières lignes permettent de voir et d'enregistrer les diverses actions. Et la grosse instruction `output.icecast` envoie la musique dans votre serveur Icecast2 en passant les paramètres nécessaires.

13.2.1 Logfile

```
set("log.file.path",   "/home/tth/Radio/log/takeover.log")
set("log.stdout",      false)
set("log.level",3)
```

Alors, la grande question est : « quelle valeur pour le loglevel ? », et la réponse n'est pas évidente à trouver dans la documentation, il y a même un ticker ouvert...

- 1 = critical
- 2 = important
- 3 = normal
- 4 = information
- 5 = debug

13.2.2 10.20.0.23

Maintenant, voyons un exemple plus complet et plus réaliste, largement inspiré d'un tutoriel de la documentation officielle, et qui deviendra un de ces

jours la base de la nouvelle [2] radio de Myrys :

```
set("log.file.path",   "/tmp/savon.log")
set("log.stdout",      false)

default = single("loveme.ogg")
openbsd = playlist(mode="random", "files/openbsd.pls")
others  = playlist(mode="random", "files/others.pls")
jingles = playlist(               "files/jingles.pls")

radio = fallback([  switch([({ 5h-11h }, openbsd),
                           ({ 11h-5h }, others)]),
                    default])
radio = random(weights=[1,12],[ jingles, radio ])

output.icecast( %vorbis,
                host      = "localhost",
                port      = 1984,
                password  = "hopla!",
                mount     = "savon.ogg",
                radio)
```

Ah bah oui, ça commence à être robuste, mais en fait c'est presque simple à comprendre, et un peu plus compliqué à mettre en œuvre.

On se retrouve vite face à un problème classique : les différences de niveau sonore entre les divers morceaux. Et comme on veut avancer sur d'autres choses, on rache un petit bout de Bash avec de la poudre verte dedans :

```
#!/bin/bash/
echo "que faire ?"
```

Cette poudre verte pourrait bien être `vorbisgain(1)`...

13.2.3 rotation des playlists

```
pl2014 = playlist("2014")
pl2013 = playlist("2013")
...
jingles = playlist("jingles")
s = rotate([pl2014, pl2013, ..., jingles])
output.icecast(..., s)
```

13.2.4 Take over the playlist

XXX

Il est également possible, avec liquidsoap, d'interrompre la diffusion d'une playlist à partir d'une autre source (fichier ou entrée audio), eventuellement externe.

2. L'ancienne ayant été fermée pour des raisons incompréhensibles.

```
#
#               TAKEOVER.LIQ
#
default = single("files/changer-monde.ogg")
liste = playlist(mode="random", "playlist.txt")
#
#  point d'entree pour le live
#
set("harbor.bind_addr", "localhost")
live = input.harbor("input", port=8088,
                        password="thegreatquux")
radio = fallback( track_sensitive=false,
                        [live, liste, default] )
output.icecast( %vorbis,
                host      = "porno.mixart-myrys.org",
                port      = 8000,
                password  = "XXXXXXXXXXX",
                mount     = "/changer-monde",
                radio)
```

13.2.5 trucs divers

Que signifie ce message : **That source is fallible.** ? Par exemple, que le fichier que vous aviez soigneusement désigné comme `default` n'existe plus sur le disque.

Comment envoyer vers plusieurs icecasts ? Bah c'est très facile : il suffit de mettre plusieurs "blocs" `output.icecast` à la suite les uns des autres ;)

13.3 Source clients

Il existe une foultitude de clients source pour abreuver icecast2.

13.3.1 ices2

Premier mystère : comment faire le rapprochement entre la déclaration du périphérique d'entrée (`<param name="device">hw:1,0</param>`) et l'équivalent dans le monde Alsa ?

La commande `arecord -L` va nous donner quelques informations sur les « points d'entrée » utilisables. Liste dont nous extrayons (par la pure pensée logique) ce fameux nom de device : `plughw:CARD=Snowball,DEV=0` XXX que nous pouvons mettre dans le XML.

Attention si votre source est monophonique, comme le microphone USB Snowball utilisé dans cet exemple, il NE faut PAS mettre *downmix* à 1...

Et si vous souhaitez dupliquer une entrée à un seul canal (genre un micro usb) por obtenir une sortie stéréo, ça ne semble pas possible fin 2020. Il serait temps de songer à écrire un patch.

13.3.2 Ezstream

> Ezstream is an audio and video streaming client which feeds any icecast server with audio data and metadata from a standard input without reencoding. It thus requires very little CPU resources. Some playlists can be looped and composed with MP3, Ogg Vorbis or Ogg Theora media files.

13.3.3 darkice

> DarkIce as a live audio streamer. It records audio from an audio interface (e.g. sound card), encodes it and sends it to a streaming server.

13.3.4 VLC

VLC est le « cône de chantier » du multimédia.

13.3.5 butt

Du clickaconvi qui plante en X remote ;(mais sinon une interface graphique simple et facile d'emploi. Un bon choix pour qui aime les choses simples où juste brancher un microphone et causer dans le Ternet.

13.3.6 idjc

> Internet DJ Console is an Internet radio application for making a live radio show or podcast. Features include two main media players with a crossfader, a jingle player, microphone signal processing (compressor and noise gate)

13.3.7 Android

Cool Mic `https://coolmic.net/` ?

Mais il faut avoir quelques connaissances en MODERNOPHONE pour arriver à tenter de comprendre pouquoi ça ne fonctionne pas, alors que tout semble correct.

13.4 Gadgets

On peut aussi imaginer des *client sources* plus ésotériques. L'exemple classique étant l'utilisation de la synthèse vocale pour faire une horloge parlante. Mais comme tout vieux linuxien, j'attache un intérêt tout particulier à une valeur symbolique : l'uptime de ma machine. En ces temps troublés [3], l'uptime,

3. Début janvier 2021, le collectif d'artistes Mixart-Myrys, dont je fais partie depuis dix ans, est en train de se faire salement éjecter de son hangartistique. À cette occasion, j'ai remonté *à la rache* un Icecast, et je l'ai voulu un peu surprenant pour le béotien, d'où l'uptime

c'est important.

```
#!/bin/bash

GS=" -g 9 -s 150 -a 133 "

#
#                Rendez-nous notre Mixou !
#
uptime -p                                                    |
        sed 's/^up/My uptime is /'                           |
        tee /dev/stderr                                      |
        espeak-ng $GS  --stdout                              |
        sox - -t wav -b 16 - remix 1 1 2> /dev/null          |
        oggenc --quiet - -o -                                |
        ~/bin/sender > /dev/null
```

L'option `-p` de la commande `uptime` permet d'obtenir une sortie de cette forme : up 1 day, 1 hour, 37 minutes qu'une micromagie `sed` change en une forme plus correcte syntaxiquement, bien qu'anglo-saxonne.

La ligne suivante sert uniquement d'affichage de contrôle, et peut être supprimée sans vergogne.

Les trois lignes suivantes vont effectivement généré le son qui sera envoyé vers la radio de l'Internet.

Premier élément du pipeline, `espeak-ng` reçoit le texte par son *stdin* et génère le flux de parole sur son *stdout*, à partir des options `-g 9 -s 150 -a 133`, réglant la voix synthétique. Ensuite, il faut que les échantillons sonores soient au format physique attendu par le diffuseur, d'où la conversion en $2x16$ bits. Et pour finir, il est encodé/encapsulé dans du OggVorbis que le *yuser* final ne pourra peut-être pas écouter sur son iPhone.

Quand à la dernière ligne `/bin/sender` c'est un programme, basé sur `libshout`, qui va gérer l'authentification et le transfert des données vers un point d'entrée du serveur icecast.

Voilà, c'est tout pour le moment.

13.5 Airtime

Cette partie est écrite pendant l'installation de Airtime dans Radio-Parasite en octobre 2021.

Hélas, ça ne va pas être si simple que ça. Le logiciel original a été abandonné depuis plusieurs années (fin 2015 à la louche) par son éditeur Sourcefabric. Un remplaçant potentiel (un fork ?) existe : *LibreTime* qui semble identique.

> LibreTime est un système d'automatisation de station de radio qui prend en charge la diffusion Web. Il est construit à partir de la communauté du système d'automatisation de la station de radio Airtime de SourceFabric.

parlé.

https://libretime.org/install

https://libretime.org/docs/host-configuration

https://progsoft.net/fr/software/libretime

13.6 libshout

Libshout c'est quoi ? Ah, chers amis de la radio en ligne, c'est une porte ouverte facilitant l'accès en direction du monde sonore extérieur, comme le précise clairement sa documentation :

> Libshout is a library for communicating with and sending data to an icecast server. It handles the socket connection, the timing of the data, and prevents bad data from getting to the icecast server.

```
/*
 *                microcast.c
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <shout/shout.h>

int main(int argc, char *argv[])
{


return 1;
}
```

Mais il reste encore pas mal de chose importante à rajouter dans ce logiciel : par exemple la partie encodage.

Chapitre 14

Backups

14.1 Geb...

Si tu fais tous tes tgz d'un coup, puis les copie, tous d'un coup, les copier juste après les avoir générés permettrait d'éviter de les relire (et potentiellement de paralléliser)

Si les fichiers sont gros (trop pour tenir en ram), et changent tous les jours, scp à la place d'rsync sera vraisemblablement plus performant (pas de comparaison de checksum sources / destinations, d'autant plus important si c'est sur des répertoires entiers)

Je suppose que tu n'utilises pas rsync -z ? Sinon, il ne sert vraisemblablement à rien, les fichiers étant déjà compréssés.

Si tu gzip plusieurs Go à chaque fois, et a plusieurs cores, pigz (https ://zlib.net/pigz/) devrait etre plus performant que gzip (tu peux le ln -s / dpkg –divert à la place de gzip), le gain est quasi linéaire par rapport au nombre de cores.

gzip, comme pigz permettent de régler le niveau de compression. Généralement diminuer celui-ci raisonnablement impacte peu la taille des fichiers générés mais énormément les temps d'exécution (et peut être la mémoire).

La réactivité de ton système s'en ressent elle si tu lances tes scripts à coup de nice -n10 (/ -n15 / -n20) sans que cela augmente trop les temps de backup ?

Quid de juste rsync sur ton serveur et faire les tgz à l'autre bout (tu profiteras ainsi pleinement du coté incrémental d'rsync) ?

14.2 rsync

A fast, versatile, remote (and local) file-copying tool.

14.3 Divers

https://changelog.complete.org/archives/10160-how-why-to-use-airgapped-backups

Perhaps surprisingly, `tar` in listed incremental mode can solve this problem for non-ZFS users. It will keep a local cache of the state of the filesystem as of the time of the last run of tar, and can generate new tarballs that reflect the changes since the previous run (even deletions). This can achieve a similar result to the ZFS send/receive, though in a much less elegant way.

Chapitre 15

Processing

Processing est tout particulièrement adapté à la création plastique et graphique interactive et au graphisme de données, mais sa simplicité d'utilisation en fait aussi un bon support pour l'apprentissage de la logique de la programmation pour les non-programmeurs.

Processing est écrit en Java, et est, en fait, une sorte de surcouche dont le principal intérêt est de masque la complexité de ce langage. Une opinion toute personnelle. Mais un outil digne d'intérêt.

15.1 Installation

Pour commencer, allez donc chercher le logiciel dans `http://processing.org/download`. Ensuire regardez ce qu'il y a dedans :

```
tth@fubar:~/$ tar ztf ~/Downloads/processing-3.5.4-linux64.tgz | head
processing-3.5.4/
processing-3.5.4/processing
processing-3.5.4/tools/
processing-3.5.4/tools/MovieMaker/
```

La documentation officielle nous explique comment installer le tarball dans votre $HOME. Personnellement, je n'aime pas cette solution, et je préfère nettement une installation *systemwide*, afin que tout les yusers y aient accès.

Nous allons donc installer l'archive dans un répertoire adapté pour les logiciels externes : `/usr/local/` [1], puis utiliser un lien symbolique pour lancer Processing.

```
tth@fubar:~$ su -
Password:
root@fubar:~\# cd /usr/local/
root@fubar:/usr/local \# tar zxf /home/tth/Downloads/processing-3.5.4-linux64.tgz
root@fubar:/usr/local \# ln -s /usr/local/processing-3.5.4/processing bin/processing
root@fubar:/usr/local \# logout
tth@fubar:~$
```

1. Certains peuvent préferer `/opt/`, mais c'est vous qui voyez...

Voilà, processing est maintenant prêt à être utilisé, modulo quelques soucis avec les items du menu "Help", que j'espère bien comprendre en plongeant dans une *stacktrace* de l'enfer.

15.2 Premiers pas

Nous allons créer une fenêtre et dessiner quelque chose dedans, ce qui semble raisonnable pour débuter...

15.3 Vers l'extérieur

Open Sound Control, aka OSC (voir page 48) pour causer sur le réseau, rs232 pour causer avec un Arduino,

15.4 Extensions

Comment peut-on écrire une extension ? Et envisageons le pire, utiliser une bibliothèque écrite en C ?

Chapitre 16

Vidéo

Le traitement de la vidéo, c'est vraiment compliqué : entre les multiples *codec* et les multiples *containers*, on est rapidement perdu. Voici donc quelques recettes de cuisine à appliquer avec modération (et respect des gestes barrières).

16.1 mplayer

Une belle grosse usinagaz...

`mplayer tv:// -tv driver=v4l2:width=640:height=480:fps=25:device=/dev/video0`

`mencoder tv:// -tv driver=v4l2:width=640:height=480:fps=25:device=/dev/video0:forceaudio:alsa=1:adevice=default -ovc lavc -oac mp3lame -lameopts cbr:br=64:mode=3 -o`

Pour faire une planche contact d'une vidéo : `mplayer -benchmark -nosound -quiet -zoom -vf scale=320:-3 -vo jpeg:outdir=/tmp -sstep 60 monfichier.ts`

16.2 ffmpeg

D'après certaines rumeurs des intertubes, ce serait un peu le remplaçant de mencoder pour pas mal de trucs, et d'après d'autres rumeurs (je confirme clairement) c'est vraiment tortueux à utiliser.

Pour convertir une sextape prise avec un *iMachin* :

`ffmpeg -i tthsf.mov -target film-dvd sextape.mpeg`

Extraire le son d'une vidéo Youtube :

`ffmpeg -i myvideo.flv -f mp3 -vn -acodec copy myvideo.mp3`

`ffmpeg -i fichiervideo.flv -vn -ar 44100 -ac 2 -f wav fichierson.wav`

Extraire les images individuelles, dans le but avoué de les maltraiter avec d'ignobles logiciels :

`ffmpeg -i foo.avi -r 1 -s WxH -f image2 foo-%03d.jpeg`

J'avoue ne pas avoir tout saisi : l'option `-r N` qui fixe le *frame rate* sert à quoi ? ? ?

Il existe bien une alternative, mais je n'ai pas encore essayé : `mplayer -vo png -ss 3:46 -frames 20 stairs.mp4` # Write out 20 png format frames from a video starting at 3 minutes, 46 seconds.

Tu peux générer les images de transition avec ImageMagick : `convert a.png b.png -alpha set -channel A -evaluate set 10% +channel -compose src-atop -composite -alpha off out.png`

Tu peux assez facilement insérer ça dans une boucle et piper les images à ffmpeg :

```
for image in ...; do
  next=... dépend de tes noms de fichiers
  for a in {0..24}; do
    convert $image $next -alpha set -channel A -evaluate set $[i*4]% \
    +channel -compose src-atop -composite -alpha off png:-
  done
done |
ffmpeg -f image2pipe -c png -i - ...
```

16.2.1 quelques trucs en vrac

```
fichier .avi -> mpeg ( qualité DVD )
    ffmpeg -i fichier.avi -target pal-dvd fichier.mpeg

fichier .avi -> mpeg ( qualité DVD, bitrate 7Mbps, 16/9 )
    ffmpeg -i fichier.avi -target pal-dvd -aspect 16:9 -b 7000000
fichier.mpeg

fichier .avi -> mpeg ( qualité dvd, en 2 passes, 16/9 )
    ffmpeg -i fichier.avi -target pal-dvd -aspect 16:9 -pass 1
fichier_tempo.mpeg
    ffmpeg -i fichier.avi -target pal-dvd -aspect 16:9 -pass 2
fichier_final.mpeg

préciser le "framerate" ( images par secondes : ici 25 )
    ffmpeg -i fichier.avi -r 25 fichier.mpeg

préciser la taille ( défaut = taille du source ( option -s ))
    ffmpeg -i fichier.avi -s 720x576 fichier.mpeg

les deux précédents :
    ffmpeg -i fichier.avi -r 25 -s 640x480 fichier.mpeg

préciser le coef de qualité ( qscale : 1 = le mieux / 3 = correcte )
    ffmpeg -i fichier.dv -target pal-dvd -qscale 2 Fichier.mpeg
```

https ://engineering.giphy.com/how-to-make-gifs-with-ffmpeg/

```
ffmpeg  -ss 0:5:59.600 -to 0:6:11.150 -i Downloads/foo.mp4 /tmp/tragic.gif
```

16.2.2 Encoder un cloître

Là, nous sommes dans du *serious bizness*, puisque cette vidéo a été programmée par James Lewis pour la résidence « Sonoptic 2020 », vidéo dont il a composé la bande originale. C'est donc un exemple réel, avec même des métadonnées.

```
ffmpeg  -nostdin                                        \
        -y -r 30 -f image2 -i lf/%06d.png               \
        -metadata artist='--[ tTh ]--'                  \
        -metadata title='-- Un cloitre en POVray --'    \
        -c:v libx264                                    \
        -pix_fmt yuv420p                                \
        -tune film                                      \
        wip.mp4
```

OMFG ! Il manque le son.

16.2.3 Capturer une vidéo

```
read -p "*** Press Enter to start recording ***"
"$FFMPEGBIN"                  \
      -thread_queue_size 512  \
      -r "$RATE"              \
      -f v4l2                 \
      -video_size "$SIZE"     \
      -i "$VIDEODEVICE"       \
      -crf 0                  \
      -c:v libx264            \
      -preset ultrafast       \
      -threads 4              \
      "$FILENAME"
```

C'est un peu rudimentaire, mais il doit être possible de *tweaker* ça avec d'autres options, genre fixer la durée de la capture.

16.2.4 Questions

Comment convertir lors de l'encodage un fichier `.WAV` en un format compressé afin de ne pas saturer la buvette ? il dit qu'il ne comprend pas la question.

Quand on encode en // images et son, comment arreter quand on atteint la fin du plus court des deux flux ? *look up the options -shortest and -apad*

Comment limiter le nombre de *cores* utilisés lors d'un encodage violent sur une toute petite machine [1] ?

ffmpeg -f x11grab -video_size cif -framerate 25 -i :0.0 /tmp/out.mpg

1. Coucou le défunt Vaio :)

16.3 VLC

VLC est un peu le couteau suisse à 112 lames du multimédia. Son emblème est un cône de chantier, je ne sais pas si ça a un sens caché. Mais soyons sérieux deux minutes. VLC propose une interface *clickconvi*, que nous allons soigneusement éviter, pour nous concentrer sur la ligne de commande.

Pour balancer toutes les vidéos en boucle random : `vlc -f -Z *.mp4`. Mais comment supprimer l'affichage du nom de fichier au début de chaque séquence ?

VLC peut éventuellement aussi être utilisé comme un outil de streaming, voyez ça page 69, bon courage.

16.4 Mediainfo

> MediaInfo [2] is a utility used for retrieving technical information and other metadata about audio or video files.

Et avec du **Zenitram** dedans *:)*

16.5 MPV

> mpv is a media player based on MPlayer and mplayer2. It supports a wide variety of video file formats, audio and video codecs, and subtitle types. Special input URL types are available to read input from a vari- ety of sources other than disk files. Depending on platform, a variety of different video and audio output methods are supported.

Un avantage de Mpv sur ses camarades (Mplayer, Ffplay...) est évident dès que l'on joue une vidéo en boucle : la transition entre la fin et le début est propre.

16.6 Autres trucs

La video informatique, c'est d'une complexité effarante. La multitude de codecs, la foultitude de containers, et les nombreuses variantes de combinaisons codec/container rendent la compréhension de cet univers assez perturbante.

16.6.1 rtmp

```
    flvstreamer -r rtmp://lacantine.live-fstreams.cdn02.rambla.be:1935/live-lacantine/live-laca
| mplayer -
```

2. by the famous Zenitram of DaLFP.

Chapitre 17

Graphisme

Voici quelques notes en vrac sur divers logiciels qui permettent de faire des dessins et des images.. . .

17.1 Flydraw

Comment faire des dessins en écrivant des lignes de codes ?

> flydraw is a drawing tool based on an interpreter which parses a language featuring rich graphic primitives ; it uses libgd to output GIF, JPEG, or PNG files.

```
comment Juste un petit dessin...

new    640,480

line 10,10,300,300,red
line 12,10,302,300,red
line 14,10,304,300,red

output  flydraw.gif
```

Pour le moment, les résultats ne sont pas encourageants :

```
tth@fubar:~/Documents/TetaTricks/code$ flydraw < dessiner.fly
bad_parms -100000
tth@fubar:~/Documents/TetaTricks/code$
```

Mais en s'obstinant un peu au *cargocult*, il s'avère que la documentation et le code semble ne sont pas d'accord. Et les messages d'erreur trop cryptiques pour moi. Donc, on y arrive.

Ceci dit, une lecture attentive du code source [1] permettra de contourner ces petits désagréments, et de rédiger quelques exemples d'utilisation amusants.

1. UTSL !

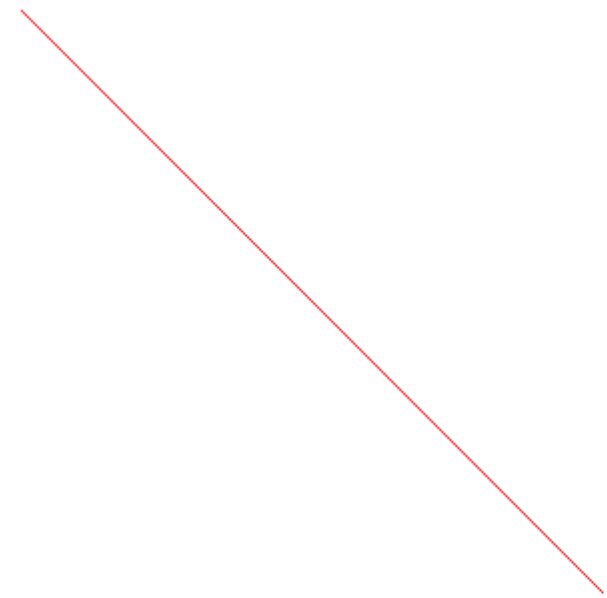


FIGURE 17.1 – Flydraw

17.2 Povray

Persistence Of Vision est un logiciel de raytracing qui existe depuis plus de trente ans. C'est un lointain descendant de DKBtrace. Mais le raytracing, c'est quoi ?

17.2.1 Caméra

Vous avez des problèmes pour caler les mouvement de la caméra ? Vous avez deux trajectoires (mal) calculées pour la `location` et le `look_at`, et rien ne tombe en face de l'objectif ? Voici une approche de solution.

La première étape est l'enregistrerement des paramètres de la caméra lors du calcul de la séquence :

```
#if (0 = clock)
```

```
  #fopen CL "cam.log" create
#else
  #fopen CL "cam.log" append
#end
#write (CL, clock, " ", CamX, " ", CamY, " ", CamZ,
                   " ", LatX, " ", LatY, " ", LatZ,"\n")
#fclose CL
```

Et la seconde étape, l'exploitation, ou plutôt la visualisation de ces données avec l'indémodable gnuplot :

```
#!/bin/bash

FILE="cam.log"
TMPF="/dev/shm/cam.data"
IMAGE="graphe.png"
tail -360 ${FILE} | sort -n > ${TMPF}
gnuplot << __EOC__
set term png    size 720,720
set output      "${IMAGE}"
set ytics       2
set xtics       2
set grid        front
set title       "Rush Camera"
plot            "${TMPF}"  using 2:4 title "Cam",\
"${TMPF}"  using 5:7 title "Lat" with dots
__EOC__
display "${IMAGE}" &
```

Dans ce cas, il s'agit d'une animation de 360 images, ce qui explique la commande`tail` au début du script. Ensuite les deux clauses *using 2 :4 / 5 :7* permettent de sélectionner les deux coordonnées X et Z, dans le repère POVray, qui n'est pas le même que celui de Gnuplot.

17.2.2 image_map

XXX

17.2.3 height_field

Les champs d'altitude... Toute une histoire...

```
#declare Champ_Altitude = object
{
height_field {
        png "WS/hf5.png" gamma 1.8
        }
pigment { color Cyan }
scale 2
}
```

Le paramètre `gamma` est injustement méconnu.

17.2.4 Random

C'est un peu différent de ce à quoi les programmeurs « classiques » sont habitués [2]. Avant de pouvoir obtenir des nombres aléatoires, il faut créer et initialiser un générateur :

```
#declare RND = seed(13.37);
```

Vous noterez au passage le fort degré alcolique de ce délicieux Corbière que je déguste en ce moment même. Ensuite, pour obtenir le résultat du tirage, on utilise le *handler* que l'on vient d'obtenir :

```
#local foo = rand(RND);
```

Par contre, on va obtenir un résultat bien classique : *The numbers are uniformly distributed, and have values between 0.0 and 1.0,*

Pourquoi nous faut-il plusieurs générateurs ? La réponse est dans la documentation [3] : *Multiple random generators are very useful in situations where you use rand() to place a group of objects, and then decide to use rand() in another location earlier in the file to set some colors or place another group of objects. Without separate rand() streams, all of your objects would move when you added more calls to rand(). This is very annoying.*

Ce qui nous conduit vers une autre question. Les PRNG sont des machines déterministes. C'est à dire qu'à conditions identiques au départ, nous aurons toujours la même séquence en sortie.

...

17.2.5 Trucs à voir

```
#declare VIRGINIA =
transform {
matrix < 1, 1, 0,
       0, 1, 0,
       0, 0, 1,
       0, 0, 0 >
       }

box { 0, 1
transform {VIRGINIA}
}
```

17.3 Avec Fortran

plplot est la piste à explorer.

2. Voir le random du C page 26
3. `https://wiki.povray.org/`

Chapitre 18

Gnuplot

Gnuplot - an interactive plotting program

There is an on-line demo collection at http ://gnuplot.info/demo

```
gnuplot> splot "cam.indoor" using 2:4:3, "cam.indoor" using 5:7:6
```

18.1 Exemple : le phytotron

Un exemple réel, issu d'un projet d'élevage d'algues bio-luminescentes dans un frigo bricolé par une jeune stagiaire fort sympathique.

Le fichier des données contient cinq champs séparés par des espaces ou des tabulations. Le premier champ est un *timestamp* exprimé en secondes depuis l'*epoch*, et les quatre suivants sont des températures en degrés Celsius.

```
#!/bin/bash
#                       THIS IS A KLUDGE

nbsamp=3000     # nombre d'echantillon

if [ $# -eq 1 ]; then
        nbsamp=$1
fi

DATAFILE="serial/foo.dat"
IMAGE="graphe.png"
TMPFILE="/dev/shm/tmpdata"
tail -${nbsamp} < ${DATAFILE} > ${TMPFILE}

gnuplot << __EOC__
set term png    size 1600,800
set output      "${IMAGE}"
set ytics       2
set xtics
set grid        front
set title       "* Temperatures du Phytotron *"
set xdata       time
```

```
set timefmt     "%s"
set format x    "%b %d\n%H:%M"
set yrange [ 10.0 : 40.0 ]
plot    "${TMPFILE}" using 1:3 title " inside" with lines, \
        "${TMPFILE}" using 1:4 title "ambient" with lines
__EOC__
```

Bon, un de ces jours, il faudrait que je ressorte des archives l'enregistrement d'un été dans le dd2, pour que vous puissiez voir le résultat de ce script en « pour de vrai ».

18.2 Paramétrer la plume

Largeur, couleur, odeur, toussa.

18.3 La 3D avec `splot`

gnuplot> help set view

The 'set view' command sets the viewing angle for 'splot's. It controls how the 3D coordinates of the plot are mapped into the 2D screen space. It provides controls for both rotation and scaling of the plotted data, but supports orthographic projections only. It supports both 3D projection or orthogonal 2D projection into a 2D plot-like map.

18.4 Des trucs...

Comment générer un signal carré :

```
set samples 2000
odd_h(x, n) = sin(x * (2*n - 1)) / (2*n - 1)
plot sum [n=1:20] 4/pi * odd_h(x, n)
```

18.5 Questions

Supposons que nous ayons sous le coude un fichier contenant *six cent mille* lignes de données, qui ne représentent que trente secondes d'enregistrement d'une valeur de Virgo, comment faire pour plotter les échantillons N à $N + 42$?

for example `plot 'a.dat' every 2`.

Chapitre 19

Webcam

Sous Linux, c'est quand même un peu tordu [1], dans le même genre que la pile son, quoi... Nous allonc commencer par les principes et outils de base, puis passer à plus simple : juste tenter de prendre une photo avec une *webcam* classique en USB, et d'enregistrer cette image.

19.1 v4l2

Video For Linux, second edition.

```
tth@delirium:~/Essais/Fonderie$ v4l2-ctl --list-formats-ext
ioctl: VIDIOC_ENUM_FMT
        Type: Video Capture

        [0]: 'MJPG' (Motion-JPEG, compressed)
                Size: Discrete 640x480
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 320x240
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1920x1080
                        Interval: Discrete 0.033s (30.000 fps)
                Size: Discrete 1280x720
                        Interval: Discrete 0.033s (30.000 fps)
```

Question : quelle méthode utiliser pour avoir la même chose avec des lignes de code écrites en C ? Il faut passer par une suite d'appels `ioctl`.

19.2 vgrabbj

D'après la manpage : « *vgrabbj is a program that will grab images from any v4l-capable device which supports one of the rgb24, rgb32, yuv420, yuv420p, yuyv, or yuv422 palettes and saves the image as a .jpg, .png, or .pnm file.* »

1. Avis personnel !

Quelques options à connaitre : `-S` échange des deux composantes R et B ; `-s <device>` affiche les capacités du périphérique v4l.

19.3 Cheese

> cheese uses your webcam to take photos and videos, applies fancy special effects and lets you share the fun with others. It was written as part of Google's 2007 Summer of Code lead by daniel g. siegel and mentored by Raphaël Slinckx. Under the hood, Cheese uses GStreamer to apply fancy effects to photos and videos. With Cheese it is easy to take photos of you, your friends, pets or whatever you want and share them with others. After the success of the Summer of Code, the development continued and we are still looking for people with nice ideas and patches ;)

19.4 Kinect

Bon, voilà un nouveau gadget ;)

À priori, la partie caméra RGB est gérée classiquement par v4l2, mais le coté spécifique, je ne sais pas qu'en faire...

Chapitre 20

X11

L'écosysteme X11 semble être en fin de carrière pour plusieurs raisons : sa complexité *mille-feuilles* accumulée au fil des décennies, ses failles de sécurité réelles ou potentielles, la difficulté de sa maintenance ; raisons qui trainent depuis trop longtemps...

À l'heure actuelle (Octobre 2023) le successeur/remplaçant du vénérable X11 « `Wayland` » commence tout juste à arriver dans les distributions Linux et à la page 92. Mais il reste un long chemin à parcourir pour une transition sans larme.

20.1 Les fontes

Comment fait-on pour connaitre la liste des polices installées ?

```
$ fc-list | awk -F ":" '{print $1}' | sort -u
```

Comment faire pour visualiser la plénitude des formes de mes fontes ?

```
tth@debian:~/Devel/TetaTricks$ man fontimage
No manual entry for fontimage
See 'man 7 undocumented' for help when manual pages are not available.
tth@debian:~/Devel/TetaTricks$
```

Wtf ?

20.2 Xephyr

D'après la manpage : *Xephyr is a kdrive server that outputs to a window on a pre-existing "host" X display. Think Xnest but with support for modern extensions like composite, damage and randr. Unlike Xnest which is an X proxy, i.e. limited to the capabilities of the host X server, Xephyr is a real X server which uses the host X server window as "framebuffer" via fast SHM XImages.*

Ça peut sembler attirant à première vue, mais l'utilisation n'est pas si simple que ça. Par exemple, quelle est la méthode à mettre en œuvre pour avoir le clavier en Azerty ?

XXX

20.3 Numlock

Vous l'avez voulu, le voici :

```
#include <X11/XKBlib.h>
#include <X11/extensions/XKB.h>
#include <X11/keysym.h>
int main(){
     Display *disp = XOpenDisplay (NULL);
     if(disp == NULL) return 1;
     unsigned int nl_mask = XkbKeysymToModifiers           \
                                  (disp, XK_Num_Lock);
     XkbLockModifiers (disp, XkbUseCoreKbd, nl_mask, nl_mask);
     XCloseDisplay (disp);
return 0;}
```

Mais c'est du très vieux code, ça !

20.4 xdotool

D'après la manpage : *xdotool lets you programatically (or manually) simulate keyboard input and mouse activity, move and resize windows, etc. It does this using X11's XTEST extension and other Xlib functions.*

En gros, et pour faire simple, avec ce logiciel, vous pouvez piloter des applications X11, à la manière d'un wm, et ce, à partir de la ligne de commande.

Vous pourrez les déplacer, les redimensionner, leur envoyer des évènements clavier, tout ce genre de choses. Il est donc possible de coder « des sortes d'animations » uniquement basées sur le protocole de X11. *Big win, bro !*

La démarche d'utilisation n'étant pas immédiate, nous allons commencer par un petit exemple : déplacer un Xlogo sur l'écran. Et comme un petit script vaut mieux qu'un long discours :

```
#!/bin/bash

xlogo &
sleep 1
job=$!    ;   echo "job␣=␣" $job

WIN=$(xdotool search -name 'xlogo')
echo "win␣=␣" $WIN

for foo in $(seq 10 33 900)
do
        sleep 1
```

```
        xdotool windowmove $WIN 50 $foo
done

kill $job
```

La première étape est le lancement de la cible en arrière-plan. La seconde est la récupération de l'indentifiant (interne à X11) de la fenêtre concernée.

```
$ xlogo &
[3] 2674
$ WIN=$(xdotool search -name xlogo)
$ echo $WIN
69206019
```

Voilà [1], nous savons maintenant à quelle fenêtre causer : par son numéro d'identification. Le problême des multiples fenêtres du même nom sera abordé ultérieurement. Nous pouvons maintenant déplacer notre widget avec vigueur...

```
$ xdotool windowmove $WIN 10 10
```

C'est simple, non ?

20.5 Le Windowmanager

Aka **WM**.

20.6 XCB

Vous avez pratiqué la *Xlib* ? Avez-vous aimé ? [OUI] et [NON] sont deux réponses valables. Et si vous ne savez pas quoi répondre, c'est que vous n'avez pas compris la question.

> The X protocol C-language Binding (XCB) is a replacement for Xlib featuring a small footprint, latency hiding, direct access to the protocol, improved threading support, and extensibility.

20.7 Xforms

À première vue [2], un toolkit sympa, très abordable en C, mais le premier exemple du tutorial d'initiation ne fonctionne pas vraiment comme il devrait. Mais quand même une affaire à suivre.

```
#include <forms.h>
/*
 *              gcc yesno.c -lforms -O YESYNO
 */
```

1. En français dans le texte.
2. Avril 2021

```
int main(int argc, char *argv[]) {
    FL_FORM *form;
    FL_OBJECT *yes,
              *no,
              *but;

    fl_initialize(&argc, argv, "Yes␣or␣No␣?", 0, 0);

    form = fl_bgn_form(FL_UP_BOX, 320, 120);
    fl_add_box(FL_BORDER_BOX, 160, 40, 0, 0,      \
                            "Do␣you␣want␣something?");
    yes = fl_add_button(FL_NORMAL_BUTTON, \
                            40, 70, 80, 30, "Yes");
    no  = fl_add_button(FL_NORMAL_BUTTON, \
                            200, 70, 80, 30, "No");
    fl_end_form();

    fl_show_form(form, FL_PLACE_MOUSE, FL_TRANSIENT, \
                             "Vroum␣!");

    while (1) {
        if (fl_do_forms() == yes) {
            printf("Yes␣␣is␣pushed\n");
            break;
        }
        else printf("␣No␣␣is␣pushed\n");
    }

    fl_finish();
    return 0;
}
```

Compilation : `gcc yesno.c -lforms -o yesno`

En fait, je ne me souviens plus du petit souci qui m'a fait négliger trop lontemps ce discret toolkit graphique, mais ça avait quelque chose à voir avec la constante `FL_BORDER_BOX` qui **ne** fait **pas** le borderbox.

http ://xforms-toolkit.org/examples.html

Chapitre 21

Wayland

L'avenir sans X11 ?

Chapitre 22

Fortran

> Fortran (FORmula TRANslator) est, avec le Cobol, un des premiers langages évolués. Il a été conçu après la seconde guerre mondiale, en 1954, par John Warner Backus, ingénieur en radiophonie chez IBM,

Pourquoi j'ai redécouvert un langage de ma jeunesse, et pourquoi vous devriez vous aussi y jeter un regard attentif. Il y a plein de bonnes choses, la septième va vous tenir compagnie.

22.1 On the big irons

TODO

22.2 FORTRAN77

Dernière version à exiger le format '80 colonnes' des cartes perforées, cette version est aussi (aux yeux de vieux dinos) la plus pure, à cause d'instructions comme le *computed goto* ou le mythique *comefrom*[1]. Nom officiel : `ANSI X3.9-1978`.

L'incarnation `g77` n'existe plus, meis peut être à-priori émulable en jouant avec les options de `gfortran` et un brin de shell.

22.3 Gfortan

Puis ce langage du passé a subi une évolution lente vers quelque chose de bien plus moderne : le Fortran 90. Les dinos vont devoir s'adapter. Et songer à utiliser un `gfortran` moderne, un élément de la Gnu Compiler Collection.

Comme pour tous les langages de programmation, il est obligatoire d'avoir l'exemple canonique, donc le voilà. Il faut l'enregister dans un fichier nommé

1. Keyword : Datamatic

`hello.f90` pour que le compilateur puisse admettre que c'est écrit en *free form format*.

```
$ cat hello.f90
program hello
    implicit none
    print *, "hello world"
end
$ gfortran -Wall hello.f90 -o hello && ./hello
 hello world
$
```

22.4 Args & Env

Un fort beau programme que ce HELLOWORLD, mais il est un peu fermé au monde extérieur. Quand on lance un programme, il peut recevoir des instructions par (mais pas que) deux canaux : les arguments de la ligne de commande et les variables d'environnement.

```
program rum_me
  implicit none
  integer               :: nbarg, foo
  character(len=32)     :: arg

  nbarg = IARGC()
  if (nbarg .GT. 0) then
    do foo=1, nbarg
       call GETARG(foo, arg)
       print *, foo,"   ",  arg
    enddo
  endif
end
```

Si un des arguments doir être vu comme une valeur numérique, il faut convertir la chaine de caractères avant usage. Exemple pour un nombre flottant :

```
  character(len=89)             :: arg
  real                          :: cx
  call getarg(2, string)
  read (string, *) cx
```

22.5 Tableaux

Rank, Size, ALLOCATABLE, toussa...

Amies du C, soyez d'entrée prévenues, en Fortran, l'indice par défaut du

premier élément d'un tableau est **1**, mais cette valeur peut être modifiée à la déclaration du tableau. Ce qui permet des *of-by-anynumber* du plus bel effet.

22.6 Nombres complexes

Partie imaginaire : `ipart = aimag(cmplx)`.

22.7 Les structures

Une « structure », c'est comme une petite boite dans laquelle on peut mettre des variables de différentes natures. Les accès à chacune de ces variables sont idépendants les uns des autres.

```
type t_pixrgb
  integer            :: r, g, b
  integer            :: alpha = 0
end type

t_pixrgb         :: pixel
pixel%r = 0x55
```

Nous verrons plus loin l'importance de cette notion.

22.8 Fichiers

Pour qui vient du C et de la philosophie Unix, la gestion des fichiers en Fortran est assez déroutante [2]. Il semble qu'il y ait deux concepts orthogonaux :

- Direct access *vs.* Sequential access
- Formtated *vs.* Unformated

Open, Read, Write, Inquire, toussa...

22.8.1 Ouvrir un fichier

C'est la première étape : attacher un fichier à un numéro d'unité. Il y a une foultitude d'options, nous allons commencer par les plus courantes.

22.8.2 Sequentiel texte

XXX

Pour faire simple, on accède à chaque unité de donnée l'une après l'autre, comme quand on déroule une bande 6250bpi. Ces unités peuvent être (restons

2. Par contre, pour un dino des *big irons*...

simple pour commencer) des lignes entières de texte. Dans l'ancien temps [3], le format de ces lignes devait être précisement structure, avec des largeurs de champs fixés.

22.8.3 Indexé

XXX

C'est pas pareil : d'abord les enregistrements sont tous de la même taille

22.9 Modules

Une notion essentielle, à mon avis, pour écrire des programmes au-delà de 104 lignes. Oui, j'avoue, j'ai bien changé depuis l'époque où je confectionnait des procédures *monobloc* de 1337 lignes, avec une bonne centaine de `GOTO` dedans.

Il serait bon de voir un exemple du monde réel. Ou juste un exemple basique. Mais avec des explications. Parce que ce n'est pas vraiment évident. Mais c'est un concept balaize.

Nous allons donc créer un module qui ne fait quasiment rien, ce qui limitera le nombre de pannes possibles dans la suite des opérations.

```
module dummy
    implicit none
contains
subroutine print_dummy
    print *, 'this is the dummy subroutine'
end subroutine
end module
```

Rien de bien compliqué : Un module peut être vu comme une boite qui contient (`contains`) des procédures (`function` ou `subroutine`, et qui sera utilisé avec ce genre de petit programme :

```
program t
    use      dummy
    implicit none
    print *, '=== programme de test ==='
    call print_dummy
end program
```

22.10 Random et Aléa

La dernière ligne va vous étonner.

3. F77 rulz the world

```
    integer, dimension(3) :: tarray
    integer               :: t3
    real                  :: dummy
    call itime(tarray)
    t3 = 3600*tarray(1) + 60*tarray(2) + tarray(3)
    write(0, '(A,3I3,A,I6)') "sranding: ", tarray, " --> ", t3
    call srand(t3)
    ! after initializing the random generator engine,
    ! you MUST use it for initializing the initializer
    dummy = rand()
```

22.11 Options de gfortran

Comme d'habitude avec gcc[4], les options sont innombrables. J'ai essayé d'en dégager les plus importantes, mais ce ne sont probablement pas les seules.

22.12 Images

La bibliothèque **Plplot** semble être un bon choix pour commencer, mais il existe bien d'autres options. Par exemple **g2** (vue en page 27) semble être utilisable avec les vieux F77, et possiblement avec un Fortran moderne, il faut juste un peu de RTFM.

Nous allons donc commencer par PlPlot, pour envisager ensuite les alternatives.

22.12.1 Plplot

Commençons directement par un EMC[5] qui ne va pas faire grand chose, à part initialiser la machinerie interne de Plplot qui est assez avancée. Il y a beaucoup de paramètres réglables, mais aussi des fonctions qui gèrent pour vous les valeurs « par défaut ». Voyez cette exemple comme un *boilerplate* simple ou un HelloWorld :

```
program plplotting
    use plplot
    implicit none

    ! Initializing the plotting package with a lot of
    ! default (and pertinent) values.
    call plinit ()

    ! Sets up plotter environment for simple graphs
    call plenv (-2.1, 2.1, -2.1, 2.1, 0, 0)

    ! Ends a plotting session, tidies up all the output
```

4. The Gnu Compiler Collection
5. Exemple Minimal Compilable

```
    ! files,switches interactive devices back into text
    ! mode and frees up any memory that was allocated.
    call plend ()

end program
```

Il ne se passe pas grand chose, sauf qu'on a une proposition de choisir le type de sortie. Dans la version que j'ai (XXX), on a le choix entre X-Window, PostScript mono ou couleur, Xfig, PNG, SVG, et bien d'autres dont certains, pour moi, assez ésotériques. D'autre part, avec cette méthode, il semble impossible de préciser la taille de l'image. Mais la démonstration est faite.

Il faut quand même regarder de plus près la subroutine `plenv()` (*Set up standard window and draw box*) qui initialise simplement un contexte graphique 2D somme toute assez classique. Elle prend six paramètres. Les quatre premiers sont les coordonnées flottantes de deux coins du papier, le cinquième controle l'échelle des axes, et le dernier controle le tracé des boites autour du graphique.

Nous avons les bases, commençons donc à dessiner.

Premièrement, nous allons chosir de démarrer automatiquement dans une fenètre X11, en gardant le reste de la configuration avec ses valeurs par défaut. Pour cela, il suffit de rajouter `call plsdev('xwin')` juste avant l'appel à `plinit`, Ensuite, pour mettre en évidence notre art, nous allons choisir une couleur pour l'encre (qui est d'un rouge du meilleur effet par défaut en mode "xwin"), puis écrire quelques légendes canoniques, donc inutiles, .

```
call plcol0 (15)         ! pure white
call pllab  ("Fuzzfactor", "Yoyodines", "Some nice plots")
```

Il est maintenant temps de poser un premier trait de crayon numérique sur notre feuille blanche numérique [6]. La forme de ce tracé sera donnée par une suite de coordonnées flottantes x/y stockées dans deux tableaux parallèles que nous allons immédiatement garnir de données pertinentes :

```
integer, parameter    :: lg = 500
real                  :: x(lg), y(lg)
real                  :: k
integer               :: i
k = 0.1
do i = 1, lg
    x(i) = k * sin(real(i)/3.0)
    y(i) = k * cos(real(i)/5.0)
    k = k * 1.006021
enddo
```

Une fois en possession de ces données conceptuelles, que l'on peut aisément voire comme un cheminement, nous allons demander au crayon de se promener sur notre feuille virtuelle en suivant ce chemin, et enfin de s'arreter de dessiner :

```
call plcol0 (12)
call plline (x, y)
call plend  ()
```

6. Actuellement noire, mais nous trouverons bien comment changer ça

Au passage, vous pouvez remarquer la magie du Fortran moderne : il n'est plus nécessaire de signifier la dimension des tableaux lors de l'appel de la procédure.

Dans ce mode de fonctionnement interactif, il vous faudra utiliser la touche <*enter*> pour sortir [7]. Nous verrons pas la suite comment écrire des fichiers 'image' vectoriels ou bitmap en mode *batch*. Enfin une ouverture vers les Gifs animées.

22.13 Questions en attente

- ✷ Comment déclarer une variable statique dans une procédure ?
- ✷ Pourquoi ne pas parler de DISLIN ?
- ✷ Est-il possible de causer à `libsndfile` ?
- ✷ Comment caler une chaine à gauche avec un FORMAT ?
- ✷ Is there a `-fortran` option for making comments with `boxes` + `figlet` ?

22.14 Réponses

<*Pas de réponse*>

Voilà, c'est tout pour le moment...

7. Une de mes prochaines tâche sera de regarder comment quitter avec le '**q**' canonique

Chapitre 23

scripting

XXX

23.1 Awk

Awk est un langage de programmation crée par Aho, Kernighan, and Weinberger [1].

Quelques *oneliners* de bon aloi pour vous donner une idée du langage...

```
cat                     { print $0 }    or just    1
grep pattern            /pattern/
head -n 5               NR <= 5
cut -f1                 { print $1 }
tr a-z A-Z              { print toupper($0) }
sed 's/hi/ho/g'         gsub(/hi/,"ho")
wc -l                   END { print NR }
```

Vous pouvez déja constater une certaine élégance portée par une grande simplicité logique, une sorte d'axiome de base, de méthologie très KISS : on va procéder en quelque sorte « demi-dalle par demi-dalle ».

23.1.1 Un exemple simple

À partir d'un jeu de données structurées, nous allons générer un fichier `.inc` contenant un objet pour Povray. Un cas d'école : un élément par ligne, les valeurs sont séparées par des espaces.

```
17      9       4       1.5
11      0.8     2.3     0.989
0       0        0      1
```

1. The AWK Programming Language, Alfred V. Aho, Brian W. Kernighan, Peter J. Weinberger, Addison-Wesley, 1988. ISBN 0-201-07981-X.

Ces données sont les coordonnées X,Y,Z et le rayon d'une petite collection de bubulles. Pratiquement, une représentation sommaire d'une sphère de povray [2].

Le fichier à générer est en trois partie : l'en-tête, la liste des bubulles et l'en-pied, ce qui est bien raccord avec la structure de déroulement d'un script Awk, comme nous allons le voir ici-même :

```
#!/usr/bin/awk -f

BEGIN   {
        print "#declare Bubules = object"
        print "{ \nunion  { "
        }

        {
        printf("  sphere { <%f, %f, %f>, %f }\n",       \
                              $1, $2, $3, $4 )
        }

END     {

        }
```

2. SDL keyword : `Sphere`

Chapitre 24

Arduino

24.1 Installation de l'IDE

Elle semble un peu plus *tricky* que celle de son cousin Processing.

Pour commencer, allez donc chercher le logiciel dans une version adaptée à votre système dans `http://arduino.cc/`.

24.2 Hello blinky

La première étape est évident : faire clignoter une LED

24.3 Beep Beep

La seconde étape ne l'est pas moins.

Chapitre 25

Basic

C'est probablement un des premiers langages interactifs utilisable par le peuple et les gens. Déja, à ce titre, dénonce que ce langage demande un peu de curiosité, il a traversé soixante ans d'informatique, et il existe encore. Mais pour moi, le véritable intérèt est dans la multitude et la diversité des implémentations. Des minis catatoniques des années 60 aux micros-ordinateurs 8 bits, des dizaines de versions souvent incompatibles ont vues le jour. Un nom pour citer un exemple ? « Locomotive », ça ira ?

Dans un premier temps, je vais les lister sommairement, et ensuite j'aurais une idée sur lesquels seront les plus funs à explorer, voire sauront trouver leur place dans un environnement dédié à la techno-futilité, concept très tendance dans certains milieux.

Pour en savoir plus : Dartmouth Time Sharing System

25.1 Bywater

> The Bywater BASIC Interpreter (bwBASIC) implements a large superset of the ANSI Standard for Minimal BASIC (X3.60-1978) and a significant subset of the ANSI Standard for Full BASIC (X3.113-1987) in C.

```
Bywater BASIC Interpreter/Shell, version 2.10
-----------------------------------------------
     Copyright (c) 1993, Ted A. Campbell
   for bwBASIC version 2.10, 11 October 1993
```

Le mode interactif me semble un peu rustre, mais utilisable à l'ancienne. Les messages d'erreur sont du genre BDOS ERROR ON A, SELECT. Je n'ai pas réussi à faire une sauvegarde du programme [1], ce qui limite son *utilisability*.

Heureusement, on peut utiliser son éditeur de texte favori pour écrire le code source, et le lancer en mode batch. Voici un exemple rudimentaire mais canonique :

1. J'ai renoncé au bout de 30 secondes

```
10  CLEAR
20  FOR I=1 TO 10
30     PRINT I, I*I
40  NEXT I
```

Première conclusion : affaire à suivre.

25.2 Yabasic

Première innovation, les numéros de ligne ne sont plus obligatoires.

```
for i=1 to 10
    print i, i*i
next
```

Et on peut faire des images qui clignotent :)

```
    open window 511,511
    for x=10 to 495 step 5:for y=10 to 495 step 5
            colour x/2,y/2,0
            fill rectangle x,y,x+5,y+5
    next y:next x
```

C'est vraiment dommage que l'on ne puisse mettre de gif89a dans un livre imprimé...

25.3 sdlBasic

http ://www.sdlbasic.altervista.org/main/

25.4 Brandy

> Brandy is an interpreter for BBC BASIC V, the dialect of BASIC that Acorn Computers supplied with their ranges of desktop computers that use the ARM processor such as the Archimedes and Risc PC, and is still in use on these and compatibles.

https ://www.bbcbasic.co.uk/index.html

25.5 FreeBASIC

> When used in its "QB" language mode, FreeBASIC provides a high level of support for programs written for QuickBASIC. Many programs written for QuickBASIC will compile and run in this mode with no changes needed.

https ://www.freebasic.net/

25.6 BASIC-256

> BASIC-256 is an easy to use version of BASIC designed to teach anybody (especially middle and high-school students) the basics of computer programming. It uses traditional control structures like gosub, for/next, and goto, which helps kids easily see how program flow-control works. It has a built-in graphics mode which lets them draw pictures on screen in minutes, and a set of detailed, easy-to-follow tutorials that introduce programming concepts through fun exercises.

https ://basic256.org/

25.7 Benchmark

Nous avons une liste de Basics, il faut maintenant les comparer. Il y a le très classique (et très ancien) crible d'Eratostène, mais nous pouvons trouver mieux en fouillant dans les archives.

25.8 Et ailleurs ?

L'interpréteur `Qbasic` de Microsoft fonctionne très bien dans Dosbox : à lire en page 125. Son prédécesseur, `GWBASIC` marche très bien aussi.

Chapitre 26

R

Je n'ai certainement pas la prétention de vous enseigner les arcanes du *R project*, je suis (octobre 2023) en pleine périoe de découverte. Mais j'ai déja un projet concret en tête, et ce chapitre sera, dans un premier temps, composé de mes notes de voyage.

> R is a language which bears a passing resemblance to the S language developed at AT&T Bell Laboratories. It provides support for a variety of statistical and graphical analyses. R is a true computer language which contains a number of control-flow constructions for iteration and alternation. It allows users to add additional functionality by defining new functions.

26.1 Les bases

Le caractère > est le prompt de R, il vous invite à taper des commandes afin qu'il vous explique le sens de la vie. On peut définir des variables, leur donner une valeur et faire des calculs avec ces variables, ce qui est bien suffisant pour commencer. Démonstration :

```
> a <- 22
> b <- 7
> a ; b
[1] 22
[1] 7
> a / b
[1] 3.142857
```

Les choses semblent simples, nous avons approximé π. Les deux premières lignes affectent des valeurs aux variables a et b. La troisième demande l'affichage de ces variables, et la quatrimème fait le calcul du nombre qu'il faut apprendre aux sages..

26.1.1 Variables

Elles sont auto-vivifiantes [1], c'est-à-dire qu'elles existent dès qu'elles sont référencées, il n'y a pas de déclaration préalable à faire.

Enfin, avec R, il ne faudrait pas parler de variables, mais d'objets, lesquels contenant les variables.

Ces objets/variables peuvent être de différents types : chaines de caractères, valeurs numériques, booléens...

26.1.2 Builtins

Quelques fonctions de base, plus ou moins vaguement expliquées.

Fonction	**Signification**
foo <- 3.14159	Affectation d'une valeur à une variable
ls()	Liste les objets actuellement chargé.
help(foobar)	Avoir de l'aide sur la chose *foobar*, abréviation : ?foobar
class(pixel)	Classe de l'objet (numeric, data.frame, ...)
str(foobar)	Information sur l'objet 'foobar'

26.1.3 Scripting

R –vanilla –quiet < mon-script.R

La magie du shebang

26.2 Un cas concret

Nous disposons d'un fichier SSV (Space Separated Values) [2] provenant d'un système optico-numérique bi-dimensionnel multi-planaire (aka Webcam). Et nous voulons en tirer quelques chiffres et graphiques bien clinquants, par exemple les éventuels rapports entre le rouge et le bleu. Pour la facilité de la mise en page, nous allons utiliser un fichier contenant seulement huit lignes de données, alors qu'un cas réel en comprend quelques centaines de milliers.

Voici les quelques lignes de ce fichier :

```
X Y R G B
0 0 0.000000 0.000000 0.000000
240 0 3967.814941 1983.907471 0.000000
480 0 7935.629883 3967.814941 0.000000
720 0 11903.445312 5951.722656 0.000000
0 240 0.000000 3967.814941 7935.629883
240 240 3967.814941 5951.722656 7935.629883
480 240 7935.629883 7935.629883 7935.629883
720 240 11903.445312 9919.537109 7935.629883
```

1. Si quelqu'un a un meilleur terme, je sui preneur
2. C'est nouveau, ça vient de sortir.

Vous pouvez remarquer que les noms de colonnes sont sur la première ligne du fichier, et qu'il y a deux types de données : les coordonnées x/y qui sont des nombres entiers, et les composantes de couleur (r/g/b) qui sont des valeurs flottantes.

Il nous faut maintenant charger ces données dans le logiciel. La fonction `read.delim` est là pour ça, il faut juste lui passer les options convenantes pour notre fichier de données : signaler la présence de l'entête avec les noms de colonnes et le type du séparateur entre champs.

```
> pixels <- read.delim("foo.txt", header=TRUE, sep=' ')
> pixels
    X   Y         R        G       B
1   0   0     0.000    0.000    0.00
2 240   0  3967.815 1983.907    0.00
3 480   0  7935.630 3967.815    0.00
4 720   0 11903.445 5951.723    0.00
5   0 240     0.000 3967.815 7935.63
6 240 240  3967.815 5951.723 7935.63
7 480 240  7935.630 7935.630 7935.63
8 720 240 11903.445 9919.537 7935.63
> class(pixels)
[1] "data.frame"
> pixels$G
[1]    0.000 1983.907 3967.815 5951.723 3967.815 5951.723 7935.630 9919.537
```

Si vous avez fait cette manipulation, bravo, vous êtes maintenant un *data-scientist* reconnu. Mais il nous reste beaucoup de choses à apprendre. Le voyage n'est pas fini. Il est temps d'explorer (ou exploiter ?) un peu nos données.

26.3 Statistiques

Oui, c'est un peu pour ça que **R** a été conçu. Hélas ce n'est pas un domaine qui m'est familier, alors ce que vous allez lire doit être pris avec des pincettes aussi bien du coté des maths sous-jacentes que de la méthodologie.

```
> mean(pixels$R) ; mean(pixels$G)
[1] 5951.723
[1] 4959.769
```

26.4 Graphiques

Bah c'est pas simple, alors j'ai activé le *rach-system* et je suis arrivé à ce petit truc qui me permet de faire de la `gif89a`, donc je suis content. Vous allez voir, c'est assez simple, mais le diable se cache dans les détails, et un peu aussi dans la doc tortueuse.

26.4.1 x/y

Un des classiques du genre.

```
R --vanilla --quiet << __EOR__
    pixels <- read.delim("cumul.txt", header=TRUE, sep=' ')
    png(filename="cumul.png", width=640, height=480)
    plot(pixels\$X, pixels\$B, pch=".",
                        main="$title", sub="tTh 2023",
                        xlab="composante rouge",
                        ylab="composante bleu")
    quit(save="no")
__EOR__
```

Détaillons un peu le script : comme dans l'exemple précédent, nous chargeons nos données dans un *data.frame*. Ensuite nous spécifions une sortie vers un fichier image au format PNG, avec une taille d'image fixée. Finalement, nous traçons notre image avec l'instruction `plot` et ses nombreux paramêtres. Je vais détailler ceux qui sont utilisés dans cet exemple, et je vous rappelle que `help(plot)` est là pour vous aider.

- `pch` : type de la plume utilisée
- `main` : titre principal de la page
- `sub` : sous-titre, placé tout en bas
- `xlab` : label de l'axe des X
- `ylab` : label de l'axe des Y

Avec ça, nous pouvons déja faire quelgues diagrammes corrects [3], mais d'autres paramêtres devraient retenir notre attention, comme la gestion des marges et des couleurs, comment exporter proprement vers LaTeX, peut-on faire de la 3D, ...

26.4.2 Histogramme

hist(datas$V3, breaks=50)

26.5 Interface avec l'OS

3. Voire même on peut tenter de la gif89a

Chapitre 27

Python

27.1 Pourquoi je n'aime pas Python

Depuis des lustres, autour de moi, je vois de plus en plus de gens faire du code en Python, certains même font de choses très avancées. Et beaucoup de ces gens essayent de me convaincre que c'est un langage vraiment trop bien, trop moderne, trop dans la *vibe* du devops, battery included, toussa. Quelques uns d'entre eux vont même jusqu'à me parler d'un certain **canard** qui faciliterait bien la vie des preogrammeurs.

Je ne suis pas convaincu.

27.2 Mais quand même...

Figure 27.1 – Chronometre

Chapitre 28

trucs de dino

Je vais, dans ce chapitre, parler de choses que les moins de vingt ans ne peuvent pas connaitre. Des choses qui existaient *bien avant* le Web, et pour lesquelles on peur facilement imaginer des usages alternatifs et amusants !

28.1 inetd

Aussi connu sous le nom de « superserveur Internet », ce démon a fait ses premiers pas dans l'Unix 4.3BSD et depuis est maintenant remplacé par `xinetd`, décrit en page 59.

Inetd a pour but de gérer l'accès à des services réseaux, qui peuvent être internes ou externes. Un service interne est entièrement géré par Inetd : on y trouve par exempme daytime qui permet de connaitre l'heure qu'il est à l'autre bout du monde :

```
tth@plop:~$ grep daytime /etc/inetd.conf
daytime         stream  tcp     nowait  root    internal
tth@plop:~$ nc uind.cispeo.fr daytime
29 JUL 2014 21:10:30 CEST
```

Dans ce cas, c'est directement Inetd qui répond. Mais il sait faire une autre chose, bien plus constructive : toujours répondre à un appel venant du grand Ternet, mais maintenanr nous allons déléguer le traitement de la demande à un autre participant.

Il ne reste plus qu'à le mettre en œuvre, en réutilisant un exemple très ancien [1], et très simple à comprendre.

Voir aussi : tcpserver...

28.1.1 qotd

Le protocole « Quote Of The Day » (`qotd`) a été normalise dans le `rfc865`, écrit en 1983 par John Postel. Une rapide implémentation du moteur de quote peut être codée en Shell :

1. Août 1998, lors d'une réunion du CULTe, à Toulouse

```
#!/bin/sh
/usr/games/fortune -a -n 500
```

Maintenant, il nous faut dire à Inetd d'appeler ce moteur chaque fois qu'une connection tcp entrante arrive sur le port 17.

28.2 Finger

finger — user information lookup program

Sur le port 79/tcp.

https ://tools.ietf.org/html/rfc1288

https ://www.bortzmeyer.org/1288.html

28.3 Gopher

Sur le port 70/tcp.

> The Internet Gopher(tm) software and documentation is copyright (c) 1991, 1992, 1993, 1994 by the University of Minnesota.

Chapitre 29

CD & DVD

Vous allez bien dire : « Encore un truc de vieux... ». Oui, peut-être, mais il y a quand même pas mal d'usages assez cools pour ces technologies du passé. Nous allons en explorer quelques uns.

29.1 CD Audio

29.1.1 Riping

Le *ripage* est l'action d'extraire les pistes d'un cd audio pour les convertir en fichier numérique audio.

Un des outils utilisés est `cdparanoia`. Les nombreuses options décrites dans la manpage montrent à quel point ce logiciel est complet et compliqué.

29.1.2 Mastering

> The Disc Description Protocol (DDP) is used by disc manufacturers as input format for their glass mastering process, and thus is the format of choice used by mastering engineers when sending out audio CD masters for replication.

29.1.3 Voir aussi...

libcdio-utils

setcd - Control the behaviour of your cdrom device

29.2 DVD

Chapitre 30

Formater du texte

Manipuler du texte, afin de le modifier, de le mettre en forme, de le présenter de façon harmonieuse, tout le monde l'a déja fait, avec des outils divers et variés et des résultats parfois (ou souvent) hasardeux.

30.1 fmt et ses copains

Nous allons voir quelques outils de base, simple et efficace, mais qui ont parfois quelques limitations. Le premier est `fmt` qui permet de remettre en forme un texte.

```
tth@plop:~/Ecriture/Ordinateur$ cat texte.txt
Ce morceau de texte n'est pas
bien
formaté.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Phasellus nunc sapien, porta id pellentesque at, elementum et felis.
Curabitur condimentum
ante in metus
iaculis quis congue diam commodo.
tth@plop:~/Ecriture/Ordinateur$ fmt texte.txt
Ce morceau de texte n'est pas bien formaté.

Lorem ipsum dolor sit amet, consectetur adipiscing elit.  Phasellus
nunc sapien, porta id pellentesque at, elementum et felis.  Curabitur
condimentum ante in metus iaculis quis congue diam commodo.
```

30.2 roff

Roff est l'outil qui a permis à quelques hackers légendaire de justifier auprès de leur hiérarchie le développement d'une machine de jeu, il est donc légitime d'en parler, non ?

Après quelques minutes d'expérimentation, un premier souci arrive. Il semble

bien que `groff` ne sache pas lire les fichiers encodés en **utf8**. Il faut donc utiliser un hack avec `recode` en prenant bien soin d'utiliser une redirection vers `stdin` comme ceci :

```
< $BASENAME.txt recode utf8..latin1 | groff -Tutf8
```

D'un autre coté, c'est `roff` qui est utilisé pour formater les pages du manuel Unix. La documentation, c'est important. C'est pour ça que j'en cause page 126.

Voyons rapidement un exemple minumum compilable. XXX

30.2.1 eqn

`eqn` compiles descriptions of equations embedded within troff input files into commands that are understood by troff.

30.2.2 tbl

`tbl` compiles descriptions of tables embedded within troff input files into commands that are understood by troff.

30.2.3 pic

`pic` compiles descriptions of pictures embedded within troff input files into commands that are understood by troff.

30.3 yodl

Yodl is a package that implements a pre-document language and tools to process it. The idea of Yodl is that you write up a document in a pre-language, then use the tools (e.g. yodl2html) to convert it to some final document language. Current converters are for HTML, man, LaTeX SGML and texinfo, a poor-man's text converter and an experimental xml converter.

30.4 zoem

General-purpose macro/programming language for transforming text

Zoem is a language for transforming text. It supports the creation of light-weight mark-up languages as macro packages built on top of zoem. PUD (Portable Unix Documentation) provides two such mini-languages and is shipped with zoem. The PUD languages are for generating manpages and FAQ documents in both HTML and troff.

Chapitre 31

Maths

Les mathématiques sont un outil indispensable dans la vie de tous les jours, même si on ne s'en rend pas forcément compte. Je ne vais pas vous faire un cours complet, j'en suis bien incapable. Je vous conseille plutôt de demander ça à monsieur schmod777, bien plus qualifié que moi.

Je vais me contenter (et c'est un peu l'esprit de cet ouvrage) de répertorier les petites choses qui rendent la vie plus facile. Nous verrons des *tips* de géométrie que vous utiliserez avec profit dans vos codes Povray, par exemple. Je songe ensuite aborder certains aspects de la théorie du signal, dans le but pratique de brotcher du son.

31.1 En vrac

```
double stddev(const auto *x, int N)
{
   double sum = 0, sum_sq = 0;
   for (int i = 0; i < N; i++) {
      sum    += x[i];
      sum_sq += x[i]*x[i];
   }
   return sqrt((sum_sq - sum*sum/N)/(N-1));
}
```

Chapitre 32

threads

Ahemmm, par quoi commencer [1] ? La programmation avec les fils d'exécution est un espace rempli de chausse-trappes. Par un extrait de la page `pthreads(6)` ? Voilà, ça c'est bien :

> POSIX.1 specifies a set of interfaces (functions, header files) for threaded programming commonly known as Pthreads. A single process can contain multiple threads, all of which are executing the same program. These threads share the same global memory (data and heap segments), but each thread has its own stack (automatic variables).

Et c'est là que les ennuis potentiels débarquent : *share the same global memory*, ce qui veut dire en gros « se partagent la même mémoire globale ». Et en français de tous les jours, « ils vont se marcher sur les pieds ».

D'un autre coté, ce paradigme permet de faire des choses assez choupi, comme un « synthétiseur d'évènement», notion que noue allons bientôt contempler.

32.1 Un exemple

Nous allons partir sur un exemple peu réaliste, mais simple à comprendre. Nous aurons un seul fil d'exécution qui affichera un compteur, puis l'incrémentera. Pendant ce temps, la branche principale du programme mettra du random dans le compteur, en espérant perturber l'autre branche.

```
/*              thread-demo                                    */

#include  <stdio.h>
#include  <unistd.h>
#include  <stdlib.h>

/* ------------------------------------------------------- */
int             compteur;           /* shared variable   */
```

1. Par de l'aspirine ?

```
/* ------------------------------------------------- */
int fil(void *ptr)
{
int             foo;

for (foo=0; foo<100; foo++) {
        compteur++;
        printf("= %d\n", compteur);
        sleep(1);
        }
return 0;
}
/* ------------------------------------------------- */
int main(int argc, char *argv[])
{
int             foo;

for (foo=0; foo<20; foo++) {
        compteur = rand()%1000;
        printf("r %d\n", compteur);
        sleep(7);
        }
return 0;
}
```

Voilà, nous avons là deux *race conditions* qui peuvent avoir des effets pervers : Ils vont effectivement se marcher sur les pieds.

Dans le thread (la fonction `fil`), le compteur est incrémenté, **puis** il est affiché. Si la branche principale poke du random dans le compteur entre ces deux opérations, *paf* le chien. Et réciproquement, si le `main` pose son aléa dans le compteur, **puis** l'affiche, cet aléa peut avoir été incrémenté par l'autre partie du processus.

Un esprit pervers vient quand même modérer ces affirmations en prétendant que dans ces deux cas, les *race conditions* pourraient dépendre des optimisations du compilateur, et de l'utilisation du `volatil` pour notre compteur.

32.2 ITC

Inter Thread Communication

Imaginons une application avec un thread qui fait le travail effectif (nous le nommerons *worker*), et plusieurs autres threads qui attendent des ordres de l'extérieur (telnet, MIDI, joystick...) et qui doivent passer ces commandes au *worker*.

Comment allons-nous procéder [2] ?

Une première piste : http ://tools.ietf.org/html/rfc3117

Une seconde : 33, page 119.

2. Et c'est une vraie question !

Chapitre 33

IPC

Inter Process Communication.

Il existe deux familles bien distinctes. Nous allons donc les voir dans le plus grand des désordres, en commençant par les grands classiques.

33.1 Signal

nous allonc commencer par le plus simple mécanisme d'IPC : le signal. Et pour ça, nous allons utiliser (encore une fois ?) un exemple futile, certes, mais simple à comprendre.

Nous avons un programme qui tourne en boucle en effectuant des taches complexes (simulées pour l'exemple par une nanosieste) et nous souhaitons pouvoir lui demander d'afficher l'avancée de la chose.

```
/*                get-signal.c                */

#include  <stdio.h>
#include  <unistd.h>
#include  <signal.h>

volatile int drapeau;

void attraper(int value)
{
drapeau = 1;
}

int main(int argc, char *argv[])
{
int              foo;

printf("kill␣me,␣my␣pid␣is␣%d\n", getpid());
signal(SIGUSR1, attraper);

for (foo=0; foo<1337; foo++) {
        if (drapeau) {
```

```
                printf("count is %d\n", foo);
                drapeau = 0;
                }
        sleep(1);       /* simulate heavy computing */
        }
return 0;
}
```

Quand ce processus recevra le signal, la fonction `attraper` sera appelée de façon *asynchrone* et positionnera le drapeau. C'est dans le terme *asynchrone* que le diable a caché les détails.

```
fubar $ ./get-signal &
[1] 14001
fubar $ kill me, my pid is 14001
kill -USR1 14001
fubar $ count is 22
kill %1
fubar $
[1]+  Terminated              ./get-signal
fubar $
```

Comme on peut le constater, le principe est simple. Un signal est une interruption d'un processus qui, en dehors de sa présence, ne transfère que peu d'information. Le principe est simple, d'accord, mais la mise en œuvre l'est moins. Cet exemple peut être considéré comme obsolète, et la *manpage*[1] confime bien.

> The only portable use of signal() is to set a signal's disposition to SIG_DFL or SIG_IGN. The semantics when using signal() to establish a signal handler vary across systems (and POSIX.1 explicitly permits this variation) ; **do not use it for this purpose**.

Vous voilà prévenus, la suite bientôt...

33.2 shared memory

Mémoire partagée.

33.3 named pipe

Un « FIFO » (ou « tube nommé ») peut être vu comme une sorte de boite à lettres dans laquelle on peut glisser un message à l'intention du processus qui aura crée cette boite, avec un nom pré-défini afin que l'expéditeur du message sache où le déposer.

> A FIFO special file is similar to a pipe, except that it is created in a different way. Instead of being an anonymous communications

1. Mais qui lit le man, de nos jours ?

channel, a FIFO special file is entered into the filesystem by calling mkfifo().

Hop, une rapide démonstration. J'avais tout d'abors pensé à un exemple minimaliste, puis je me suis dit : « Et pourquoi ne pas enmbarquer une charge utile dans le message ? », aussitôt dit aussitôt fait.

L'émetteur mettra un *timestamp* dans le corps du message, et le récepteur affichera l'écart avec le *timestamp* local. Ce marqueur temporel sera obtenu par l'appel à `gettimeofday(2)` pour tenter d'avoir une précision significative.

33.3.1 Émetteur

Son role est simple : il met le timestamp dans l'enveloppe, et la poste vers l'adresse qu'on lui passe sur la ligne de commande.

```
/***             named pipe --- transmiter        ***/
#include  <stdio.h>
#include   <stdlib.h>
#include   <string.h>
#include  <unistd.h>
#include  <fcntl.h>
#include  "my-fifo.h"

int main(int argc, char *argv[])
{
int              fifo;
Message          message;

if (argc!=2)             exit(1);
if (-1==(fifo = open(argv[1], O_WRONLY))) {
        perror("open␣fifo␣for␣wr");
        exit(1);
        }
memset(&message, 0, sizeof(Message));
message.pid = getpid();
message.timestamp = dtime();
write(fifo, &message, sizeof(Message));
close(fifo);

return 0;
}
```

Vous remarquerez que nous nettoyons le message avant utilisation avec un `memset(3)`, une étape trop souvent oubliée.

33.3.2 Récepteur

Pour faire simple, le récepteur des messages va passer l'intégralité de son temps scotché devant la boite aux lettres, et attraper le message dès son arrivée. Dans notre jargon, nous appellons ça la méthode *buzzy-wait*.

Sauf que ça n'est pas tout à fait ça.

```
/***            named pipe --- receiver          ***/

#include  <stdio.h>
#include  <stdlib.h>
#include  <unistd.h>
#include  <sys/stat.h>          /* for mkfifo */
#include  <fcntl.h>
#include  "my-fifo.h"

int main(int argc, char *argv[])
{
int             fifo;
Message         message;
double          localTS;

if (argc!=2) {
        fprintf(stderr, "%s␣need␣a␣fifo␣name\n", argv[0]);
        exit(1);
        }
fifo = mkfifo(argv[1], O_RDONLY|O_CREAT);
if (-1==fifo) {
        perror("mkfifo␣fail");
        exit(2);
        }
read(fifo, &message, sizeof(Message));
localTS = dtime();
printf("%f␣␣%f\n", localTS, message.timestamp);
close(fifo);
return 0;
}
```

J'avoue ne pas avoir encore bien capté le deuxième argument de mkfifo, mais la lecture de `fifo(7)` et de `pipe(7)` peut être utile pour comprendre [2] la chose...

33.3.3 Timestamp

Comme promis un peu plus haut, le code est très simple. C'est juste dommage qu'une fonction aussi utile ne fasse pas partie de la *stdlib* du langage C, alors qu'elle le mérite.

```
#include <stddef.h>
#include <sys/time.h>
#include "my-fifo.h"
double dtime(void)
{
struct timeval tv;
gettimeofday(&tv, NULL);
return (double)tv.tv_sec + ((double)tv.tv_usec)/1e6;
}
```

2. modulo les références circulaires.

Cecit dit, la précision, ou plutôt la résolution, est très dépendante du matériel, de sa gestion par le système d'exploitation, et et aussi [3] du BOFH local.

> The time returned by gettimeofday() is affected by discontinuous jumps in the system time (e.g., if the system administrator manually changes the system time). If you need a monotonically increasing clock, see clock_gettime(2).

33.3.4 Message

Pour des raisons pas si évidente que ça, les messages transmis doivent tous avoir la même taille. Nous allons donc définir une structure (page ? ? ?) contenant les données à transmettre.

```
typedef struct {
        int             pid;
        double          timestamp;
        long            notused;
        } Message;

double dtime(void);
```

C'est rudimentaire, mais ça fera l'affaire pour claquer un Bonaldi.

3. Mais faut-il s'étonner ?

Chapitre 34

Dosbox

Dosbox est un émulateur de machine MS-DOS avec le son et le graphique. Il ne nécessite pas de système d'exploitation.

34.1 Configuration

La configuration de Dosbox est par défaut dans le fichier `$HOME/.dosbox/dosbox-0.74-2.conf`. Vous remarquerez que le numéro de version est **dans** le nom de ce fichier. Ce fichier est abondamment commenté.

C'est à la fin de celui-ci, dans la section `autoexec` que vous pouvez rajouter ce genre de lignes :

```
mount C: /home/tth/Essais/DosBox/C
mount D: /home/tth/Essais/DosBox/D
keyb fr
```

34.2 In an Out

Q : J'ai téléchargé un warez de Qbasic [1] pour jouer avec, le l'archive 7z ne contient qu'un fichier `.img`. Que puis-je faire pour jouer enfin à snake ?

Q : Il y a plusieurs réponses possibles. La plus évidente est de passer par le système hote pour lire le contenu de cette image :

```
tth@fubar:~/DosBox/QBasic-1.1 $ su -
Password:
root@fubar:~# cd ~tth/DosBox/QBasic-1.1
-bash: cd: too many arguments
root@fubar:~# cd ~tth/DosBox/QBasic-1.1 /
root@fubar:/home/tth/DosBox/QBasic-1.1 # ls
disk01.img  winworldpc.com.txt
root@fubar:/home/tth/DosBox/QBasic-1.1 # mount -o loop,ro disk01.img /mnt
root@fubar:/home/tth/DosBox/QBasic-1.1 # ls /mnt
QBASIC.EXE  QBASIC.HLP
```

1. cf. page 103

Ceci dit, il y a encore plus simple : « mounter » l'image disque afin qu'elle soit directememt utilisable depuis le prompt Dosbox. Consultons vite la doc [2], juste pour constater que c'est un peu technique.

```
IMGMOUNT DRIVE [imagefile] -t [image_type] -fs [image_format]
          -size [sectorsbytesize, sectorsperhead, heads, cylinders]
IMGMOUNT DRIVE [imagefile1 imagefile2 .. imagefileN] -t cdrom -fs iso
```

34.3 Popcorn

Popcorn est un jeu de cassebrique écrit dans les années 80 pour tirer la quintessence des cartes graphiques CGA. Écrit, sans le moindre doute, au « ras du métal ». Une des merveilles de l'ancien temps.

34.4 Qbasic

34.5 Fractint

La première chose qu'il faudrait regarder, c'est les options de la ligne de commande, pour pouvoir batcher des animations...

2. /usr/share/doc/dosbox/README.gz

Chapitre 35

manpages

man is the system's manual pager. Each page argument given to man is normally the name of a program, utility or function. The manual page associated with each of these arguments is then found and displayed. A section, if provided, will direct man to look only in that section of the manual. The default action is to search in all of the available sections following a pre-defined order ("1 n l 8 3 2 3posix 3pm 3perl 3am 5 4 9 6 7" by default, unless overridden by the SECTION directive in /etc/manpath.config), and to show only the first page found, even if page exists in several sections.

35.1 Un exemple ?

XXX

Voir `roff` en page 114.

Chapitre 36

Grub

Successeur pour certaines distributions du vénérable `lilo`[1], le Grub, aka *Grand Unified Bootloader* est nettement plus complexe, mais permet (parait-il) de faire beaucoup plus de choses.

36.1 Astuces

Pour installer `memtest86+` dans une Fedora, c'est en page **??**.

36.2 Questions

- comment passer la carte VGA en 80x43 ?
- comment booter un disque FreeDOS ?

1. Linux Loader

Chapitre 37

Photographie

Le domaine de la photographie numérique est vaste. Nous allons essayer d'en entrevoir certains aspects et de découvrir quelques recettes bien pratiques. Dans le (dés)ordre, nous verrons comment récupérer les image d'un appareil qui n'est pas compatible avec le bien pratique protocole *mass storage*, puis nous passerons au traitement des fichiers crus, les fameux RAW qui font le bonheur de monsieur Reep.

Voir aussi ImageMagick dans le chapitre 10, page 51, un bon outil pour brotcher vos images.

37.1 gphoto2

Notes écrites en se basant sur la version 2.5.20 (Debian 10.23), les choses peuvent être (rarement) différentes chez vous. L'appareil photo que j'utilise actuellement est un Canon *EOS 600D*[1], dont, il faut bien l'avouer, la complexité m'effraye un peu.

```
gphoto2 --list-ports
gphoto2 --list-cameras
gphoto2 --auto-detect
```

Ces trois commandes servent à obtenir la description des modes de connexion possibles, la liste des appareils gérés, et l'éventuel appareil connecté, si il est sous tension, bien entendu :)

```
tth@redlady:~/Photos/Canon600D/2022-12$ gphoto2 --auto-detect
Model                          Port
----------------------------------------------------------
Canon EOS 600D                 usb:002,005
```

Ok, nous savons maintenant par quel petit nom notre Canon est connu par le système sous-jacent.

```
gphoto2 --port usb:002,002 -P
```

1. Hélas, j'ai perdu mon bien-aimé PowerShot A520...

Cette commande fait une récupération bestiale de toutes les photos (l'option `-P` peut être remplacée par `-get-all-files`). L'option `-new` permet de se limiter au téléchargement des nouvelles photos.

Pour les drogués de la ligne de commande, l'option `-shell` démarre un environnement interactif, qui permet d'explorer le contenu de votre appareil, et d'interagir avec lui. Ce qu'on peut faire semble assez dépendant du modèle d'apn utilisé.

Il doit exister plein d'autres tricks'n'tips, il faut juste prendre le temps de les découvrir.

37.2 ufraw

Ufraw (Unidentified Flying Raw) est un convertisseur de fichier images *RAW*, c'est à dire en général les données brutes en sortie du capteur de l'appareil photo.

37.3 EXIF

`exiftool - Read and write meta information in files`

> **from wikipedia :** Exchangeable image file format (officially Exif, according to JEIDA/JEITA/CIPA specifications) is a standard that specifies formats for images, sound, and ancillary tags used by digital cameras (including smartphones), scanners and other systems handling image and sound files recorded by digital cameras.

37.3.1 Lire

Peut-être est-il temps de passer (comme d'habitude) à un exemple pratique. J'ai un appareil numérique [2] et quatre objectifs. Je souhaite écrire le nom de l'objectif sur l'image, et de préférence dans une teinte assez kitch. Comment faire ?

37.3.2 Écrire

Un autre exemple me vient à l'esprit : Je souhaite rajouter un petit texte dans les *tags* d'un fichier JPEG. Comment procéder ?

Et surtout sans perdre les *tags* déja présents.

2. un Canon EOS 600D, mais ça n'est pas important

Chapitre 38

Plugins

Le concept de bibliothèque partagée (*shared library* en anglais est apparu, du moins dans le monde Unix, avec *SunOS 4*, en 1989. Mais il est probable que ce concept existait déja dans d'autres mondes, peut-être avec *AmigaOS* ou *VAX/VMS*...

Une bibliothèque partagé (aka `.so`) est une collection de fonctions, chargée une seule fois en mémoire centrale, mais qui peut être utilisée par plusieurs programmes en même temps. Et c'est en partie sur ce concept qu'est basé le premier mécanisme que nous allons découvrir : `dlopen` et sa petite famille.

38.1 Mécanisme

`dlclose, dlopen, dlmopen - open and close a shared object`

> The function dlopen() loads the dynamic shared object (shared library) file named by the null-terminated string filename and returns an opaque "handle" for the loaded object. This handle is employed with other functions in the dlopen API, such as dlsym(3), dladdr(3), dlinfo(3), and dlclose().

Traduit en clair, la fonction `dlopen` permet de charger manuellement une bibliothèque partagée (un `.so`) en mémoire centrale, au lieu de laisser faire le loader (ld.so) automatiquement au lancement d'un binaire. la fonction `dlclose` va, elle, vidanger la zone mémoire dans laquelle est le plugin.

38.2 Un exemple ?

Comme d'habitude, je vais proposer un exemple un peu artificiel, mais qui, de part sa simplicité, illustrera clairement les concepts de base. Partons du principe que nous avons un tableau de quatre nombres flottants, et que nous souhaitons proposer un affichage de ce tableau modifiable à volonté.

Nous allons donc écrire notre fonction personnelle d'affichage dans un fichier .c indépendant (ci-après dénommé «le plugiciel») et qui sera chargé dynami-

quement au *run-time* par le logiciel principal (l'appelant).

38.2.1 Le plugiciel

C'est ici que nous trouverons le code actif de l'exemple, l'utilité de notre *usecase* reste à déterminer par nos lecteurs, mais seulement à titre d'exercice.

```
/*              plugiciel.c                          */

#include  <stdio.h>
#include  "plugiciel.h"

void affichage(char *titre, float vals[4])
{
printf("%s : %f %f %f\n", titre,
             vals[0]*vals[3], vals[1]*vals[3],
                   vals[2]*vals[3]);
}
```

On peut difficilement faire plus simple pour une première approche, mais attendez la suite des opérations, la cinquième va vous étonner.

38.2.2 L'appelant

Voyons maintenant comment mettre ça en œuvre depuis le programme principal. Nous supposerons que le binaire du plugin est dans le répertoire courant. Pour les autres cas, il faut jouer avec la variable d'environnement `LD_LIBRARY_PATH` (cf man dlopen(3)) ou le fichier `/etc/ld.so.cache` (cf man ldconfig(8)).

```
/*              appelant.c                           */

#include  <stdio.h>
#include  <dlfcn.h>

void affichage(char *titre, float vals[4]);

int main(int argc, char *argv[])
{
void            *handle;
float           values[4];

void (*funcname)(const char*, const float *);

handle = dlopen("./plugiciel.so", RTLD_LAZY);
if (!handle) {
        /* fail to load the library */
        fprintf(stderr, "Error: %s\n", dlerror());
        return 1;
        }

*(void**)(&funcname) = dlsym(handle, "affichage");
```

```
values[0] = 13.37;       values[1] = 16.64;
values[2] = 3.14159;     values[3] = 0.5;
funcname("rgb␣*␣a␣=", values);

dlclose(handle);

return 0;
}
```

Ah, les choses se corsent un peu, il y a un pointeur bien tortueux à déchiffrer. Il était temps.

```
void (*funcname)(const char*, const float *);
```

38.2.3 Le run

```
tth@fubar:~/Documents/TetaTricks/code$ make plugin
gcc -Wall -shared -fPIC plugiciel.c -o plugiciel.so
gcc -Wall appelant.c -ldl -o appelant
tth@fubar:~/Documents/TetaTricks/code$ ./appelant
rgb * a = : 6.685000 8.320000 1.570795
tth@fubar:~/Documents/TetaTricks/code$
```

38.3 Autres langages

En Perl ?

En Fortran ?

Chapitre 39

Bonjour

Bonjour (anciennement nommé RendezVous) est l'implémentation par Apple de la norme Zeroconf. C'est un système de mise en réseau local automatique.

Bonjour utilise le port **5353** en UDP. Pour IPv4, la RFC 3927 de l'IETF définit l'allocation dynamique d'adresses IP dans la plage 169.254.0.0/16.

Avahi est une bibliothèque logicielle fournissant une implémentation libre des protocoles IPv4LL, mDNS et DNS-SD.

39.1 Outils

Bon, maintenant, que faire de tout ça ? Explorer ce qui se trame sur notre réseau local ? C'est évident que si il y a rendez-vous, il y a forcément de la drague sur le LAN. Et la drague, le meilleur moyen c'est quand même du broadcast sur UDP. Il y a même un logiciel spécialisé pour entendre ça :

> mdns-scan is a tool for scanning for mDNS/DNS-SD published services on the local network. It issues a mDNS PTR query to the special RR _services._dns-sd._udp.local for retrieving a list of all currently registered services on the local link.

```
tth@debian:~/Devel/TetaTricks$ mdns-scan
+ saned._sane-port._tcp.local
+ DELIRIUM._smb._tcp.local
+ DELIRIUM._device-info._tcp.local
+ Serviette._nut._tcp.local
+ Serviette._mumble._tcp.local
+ RENAN-PC._nvstream._tcp.local
+ Mac Pro de XANO._afpovertcp._tcp.local
+ Mac Pro de XANO._smb._tcp.local
+ MacBook Pro de Romain._smb._tcp.local
+ Mac Pro de XANO._companion-link._tcp.local
+ HP Photosmart C4400 series @ MacBook Pro de Sophie de Angelis (._ipp._tcp.local
+ TOSHIBA e-STUDIO2040C-07304719 @ MacBook Pro de Sophie de Angel._ipp._tcp.local
+ TOSHIBA e-STUDIO2010AC-13828385._http._tcp.local
+ MacBook Pro de Manon._companion-link._tcp.local
```

```
+ Mac Pro de XANO / Mac Pro de MacPro._c4ds._tcp.local
+ 0418C7E2-071B-4BC7-B210-20F4D6BED80A._netrender._tcp.local
```

Que faire de toutes ces information, à part savoir que la machine `delirium` semble être un serveur Samba ?

39.2 Code

Je suppose qu'il faut commencer par découvrir Avahi et ses quelques fichiers de configuration [1].

- `/etc/avahi/hosts`
- `/etc/dbus-1/system.d/avahi-dbus.conf`
- `/etc/avahi/avahi-daemon.conf`
- `/etc/init.d/avahi-daemon`

Attention aux yeux, il y a parfois du XML dans ces choses là..

1. Chez Debian

Chapitre 40

I/O control

Un des paradigmes les plus connus à propos des systèmes de la famille Unix est : Tout est vu comme un fichier. C'est souvent vrai, mais parfois ce n'est pas suffisant. Prenons rapidement un exemple, le baton de joie dans Linux.

```
int                joy_fd;
char               joy_name[128];

joy_fd = open(joy_device , O_RDONLY);
if (ioctl(joy_fd, JSIOCGNAME(sizeof(joy_name)), joy_name) < 0)
        strncpy(joy_name, "Unknown", sizeof(joy_name));
fprintf(stderr, "Name: \"%s\"\n", joy_name);
```

La macro `JSIOCGNAME` étant définie dans le fichier `/usr/include/linux/joystick.h` comme `_IOC(_IOC_READ, 'j', 0x13, len)`. Bon, OK, j'aurais pu choisir un exemple plus simple, mais vous avez capté le principe...

Ailleurs dans ce document : Ports série (p. 152)

Chapitre 41

Debug

Quand plus rien ne marche, reste-il encore un espoir ? Il existe bien entendu des outils *mainstream* tels que le classique `gdb`, mais il en existe une foultitude d'autres, injustement méconnus. Nous allons en voir quelques-uns en essayant de nous baser sur des cas quasiment réels.

Et non, je ne vais pas vous parler du vénérable `ddt` des antiques systèmes CP/M, ni même du `debug` des anciens DOS, bien que tous deux méritent votre attention.

41.1 Gdb

GDB is a source-level debugger, capable of breaking programs at any specific line, displaying variable values, and determining where errors occurred. Currently, gdb supports C, C++, D, Objective-C, Fortran, Java, OpenCL C, Pascal, assembly, Modula-2, Go, and Ada. A MUST-HAVE FOR ANY SERIOUS PROGRAMMER.

41.1.1 Clickaconvi

DDD is a graphical front-end for GDB and other command-line debuggers. Using DDD, you can see what is going on "inside" another program while it executes—or what another program was doing at the moment it crashed.

xxgdb is a simple but powerful graphical interface to the GNU debugger gdb. A more powerful (but slower and much bigger) interface is available in the ddd package.

41.1.2 Un cas réel

J'ai un programme, écrit en C, qui est assez agressif sur le calcul flottant et les accès disque, et qui se fait tuer en cours de route, avec un message inquiétant du kernel `[95335.731943] fonderie: Corrupted page table at address 7fffe82d6000` qui me laisse perplexe.

```
[tth@laserbox Sauvageonnes]$ gdb ~/Devel/FloatImg/Fonderie/fonderie
GNU gdb (GDB) Fedora 10.1-2.fc33
This GDB was configured as "x86_64-redhat-linux-gnu".
Reading symbols from /home/tth/Devel/FloatImg/Fonderie/fonderie...
(gdb) run -I 'G/?????.fimg' -O 'Png' -w 0 -x 0 -T 80
Starting program:
            /home/tth/Devel/FloatImg/Fonderie/fonderie
            -I 'G/?????.fimg' -O 'Png' -w 0 -x 0 -T 80
Missing separate debuginfos, use:
            dnf debuginfo-install glibc-2.32-2.fc33.x86_64
*** /home/tth/Devel/FloatImg/Fonderie/fonderie :
        compiled by tTh, Nov 25 2020 10:19:10
        pid 1949
```

Là, on attend vingt minutes que le logiciel tripote plein d'images en virgule flottante. C'est long.

```
           54 /  1784
Program terminated with signal SIGKILL, Killed.
The program no longer exists.
Missing separate debuginfos, use:
            dnf debuginfo-install pnglite-0.1.17-1.fc33.21.x86_64
            zlib-1.2.11-22.fc33.x86_64
(gdb)
```

On est donc bien avancé, `SIGKILL`, dans ce cas-là, c'est le noyau qui flingue le processus. Mais pour quelle raison, et à quel endroit dans le programme ? Avec un peu de chance, gdb a conservé une trace des dernières microscondes de la fonderie.

```
(gdb) backtrace
No stack.
(gdb)
```

Bah non, il nous avait bien prévenu : *The program no longer exists*, point suivant.

41.2 Valgrind

Reprenons le premier exemple précédemment traité avec Gdb, et tentons de résoudre l'énigme avec Valgrind.

```
(gdb) quit
[tth@laserbox]$ man valgrind
[tth@laserbox]$ valgrind fonderie -I 'G/?????.fimg' -O 'Png' \
                                          -w 0 -x 0 -T 80
==2388== Memcheck, a memory error detector
==2388== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2388== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
```

Là, on attend vingt heures que le logiciel tripote plein d'images en virgule flottante. C'est très long. Je me demande même si ce n'est pas un roblock total.

41.3 Strace

Strace permet de tracer les appels systèmes d'un processus. Comme vous le savez tous, un appel système (aka syscall) est **le** moyen de communication qu'utilise un process utilisateur pôur demander un service au noyau.

```
#include  <stdio.h>

int main(int argc, char *argv[])
{
puts("hello␣world.");
return 0;
}
```

Un exemple canonique, n'est-il pas ? Ce bout de code affichant quelque chose à l'écran, il doit bien y avoir un appel au noyau qui traine par là pour écrire vers la sortie standard. Nous allons donc le chercher [1]

```
$ gcc hello.c
$ strace -o foo ./a.out
$ grep write foo
write(1, "Hello world.\n", 13)          = 13
$
```

On peut réaliser la même opération en utilisant le filtrage interne de strace, ce qui évite le passage par un fichier intermédiaire, mais le résultat est moins lisible, puisque a.out écrit sur stdout, et strace sur stderr :

```
$ strace -e write ./a.out
write(1, "Hello world.\n", 13Hello world.
)          = 13
+++ exited with 0 +++
$
```

41.4 LD_PRELOAD

D'accord, aves `strace` nous pouvons voir passer les appels systèmes, mais dans la vie d'un process, certaines opérations ne sortent pas de la `libc`. L'une d'entre elles, `getenv(3)`, va nous servir d'exemple.

```
NAME
       getenv - get an environment variable
SYNOPSIS
       #include <stdlib.h>
       char *getenv(const char *name);
DESCRIPTION
       The  getenv()  function searches the environment list to find the envi-
       ronment variable name, and returns a pointer to the corresponding value
       string.
```

1. En trichant un peu, je l'avoue, je connais son nom.

Cette fonction est utilisée par un logiciel pour avoir accès à son contexte extérieur, son environnement, dans lequel on peut trouver (entre autres) la variable `$LOGNAME` qui, oh surprise, contient votre nom de login.

Et justement, vous avez un programme sous la main que vous suspecter d'avoir un problème relationnel avec cette variable. Il nous faut donc remplacer le getenv de la libc par notre propre version qui va écouter et exfiltrer l'utilisation de cette fonction.

```
/*
 *
 *  spy_getenv.so:       spy_getenv.c Makefile
 *       gcc -Wall -shared -fPIC  $< -ldl -o $@
 *
 */

#include  <stdio.h>
#include  <string.h>
#include  <unistd.h>

#define   __USE_GNU
#include  <dlfcn.h>

typedef char * (*original_getenv)(const char *envname);

char *getenv(char *envname)
{
static char             *arrow = "--getenv-->␣␣";
static char             *wtf = "␣-->␣(nil)";

char                    *content;
original_getenv         orig_getenv;

orig_getenv = (original_getenv)dlsym(RTLD_NEXT, "getenv");

write(STDERR_FILENO, arrow, strlen(arrow));
write(STDERR_FILENO, envname, strlen(envname));

content = orig_getenv(envname);
if (NULL != content) {
        write(STDERR_FILENO, "=", 1);
        write(STDERR_FILENO, content, strlen(content));
        }
else    {
        write(STDERR_FILENO, wtf, strlen(wtf));
        }

write(STDERR_FILENO, "\n", 1);

return content;
}

/* j'adore quand un plan se deroule sans accroc */
```

Et à l'utilsation :

```
tth@redlady:~$ LD_PRELOAD=~/Devel/TetaTricks/code/debug/spy_getenv date
--getenv-->  TZ --> (nil)
--getenv-->  TZ --> (nil)
Tue Oct 17 08:55:01 CEST 2023
tth@redlady:~$
```

Simple et efficace.

41.5 Cflow et Slint

Deux outils pour savoir quoi nettoyer dans du code C un peu relou.

Chapitre 42

gif89a

Ah, enfin nous arrivons à un sujet **vraiment** important : la confection d'images qui clignotent. Ça va aussi être l'occasion de plonger un peu dans l'histoire du siècle dernier. Nous allons parler de Compuserve que seul les plus anciens d'entre vous on pu connaitre.

> *Wikipedia* : CompuServe (ou CompuServe Information Services, ou CIS), est le premier des grands fournisseurs de services en ligne aux États-Unis. Il domine le marché pendant les années 1980 puis reste un acteur important au milieu des années 1990.

Voilà pour expliquer brièvement d'où sort ce fameux format GIF. À l'époque, les moyens de connexion étaient beauuucoup plus lents que ce que nous connaissons en 2020 [1], mais les écrans graphiques commençaient à bien se répandre, et certains pouvaient même afficher *256 couleurs* :

> L'entreprise est aussi connue pour avoir mis au point un format d'image numérique très répandu, le GIF, en 1987.

Deux ans plus tard, le public demandant de plus en plus d'images épileptisante, une extension de ce format fut crée, et reçu le petit nom de GIF89A. Il y a toujours la limitation à 256 couleurs et un seul niveau de transparence. Contraintes qui n'étaient pas si génantes que ça à l'époque vu les terminaux visés. Contraintes qu'un artiste numérique peut comprendre et accepter.

42.1 Création

Avec la commande `convert` d'ImageMagick, on peut, à partir d'une suite d'image, créer une gif89a qui tourne en boucle :

```
convert -delay 20 -loop 0 a.png b.png c.png foo.gif
```

Le *delay* est exprimé en millisecondes. On peut aussi rajouter l'option `-loop nnn` [2] avec nnn qui est le nombre de boucle que doit faire l'animation. La valeur

1. Une fscking mauvaise année
2. aka : add Netscape loop extension to your GIF animation

0 demandant une boucle infinie. Deux fois [3].

Et ensuite, vouv pouvez voir votre logo en flamme avec la commande `animate foo.gif` sur votre super écran (s)vga 14 pouces.

Je pense [4] que la compression de cette commande n'est pas optimale, peut-être parce qu'elle n'utilise pas les *subblocs* que permet ce format.

42.2 Manipulation

XXX

42.3 Exemples pratiques

Dans un Makefile [5] :

```
indoor.gif: $(DEPS)
    convert                                             \
        `ls -rt1 frames/indoor/????.png | tail -99`     \
        -delay 120                                      \
        -resize 28%                                     \
        -normalize                                      \
        -colorspace gray                                \
        -colors 31                                      \
        -dither none                                    \
        $@
```

42.4 gifsicle

'Gifsicle' manipulates GIF images. Its most common uses include combining single images into animations, adding transparency, optimizing animations for space, and printing information about GIFs.

3. Vous connaissez Chuck Norris ?
4. Avis personnel d'après ma louche.
5. Les makefiles, c'est le bien !

Chapitre 43

Internet Relay Chat

43.1 hybrid

C'est le serveur anciennement utilisé par Smeuh. Il semble que ce soit une vrai galère à configurer.

On se retrouve vite dans des situations critiques, du genre un #canal où il n'y a plus de @op, et donc, on ne peut plus changer le topic, et là, **ça craint vraiment** !

43.2 irssi

> Irssi is a modular Internet Relay Chat client. It is highly extensible and very secure. Being a fullscreen, termcap based client with many features, Irssi is easily extensible through scripts and modules.

```
/recode add #gcu ISO-8859-15
pinpin0: çhârsêt ?

/SET autoclose_windows OFF
```

Question : comment enregistrer un chan' ?

43.3 hexchat

Client graphique bien efficace.

43.4 Les "bots"

Hello Dotanick.

Chapitre 44

Usenet

Je suis un vieux dino du grand Usenet mondial. Mes premiers posts doivent dater de quelques mois après l'arrivée de Compu$erve en France. C'est dire... Et quand il m'arrive de parler de cet ancêtre des réseaux sociaux, la question : « Usenet, c'est quoi ? » revient parfois [1]. On va essayer d'y répondre rapidement, avant de passer aux outils qui le font fonctionner.

Usenet was born approximately three decades ago, in 1979. It all began as a small communication network between a few universities in the United States used for the purposes of trading information, sharing news, discussing new and exciting developments, and collaborating on research results and projects.

44.1 slrn

> slrn is an easy to use but powerful NNTP / spool based newsreader. It is highly customizable, supports scoring, free key bindings and can be extended using the embedded S-Lang interpreter.

```
printf 'set wrap_flags 7\nset wrap_width 72\n' >> .slrnrc
```

44.2 leafnode

Leafnode est l'outil essentiel si on désire troller au fond du jardin, en dehors du rayon d'action de sa box. On peut le voir comme une sorte de proxy, mais pas que. Il permet aussi d'agréguer plusieurs feeds, ce qui permet de combler les trous, et de donner accès à Usenet à votre réseau local.

44.3 INN

Voilà, c'est le moment de parler de l'engin fondamental de ce réseau de discussions. INN.

1. Mais pas assez souvent, à mon gout...

44.4 fufe

N'y allez pas, **It's a trap !**

Chapitre 45

Domain Name Service

Le DNS est (pour simplifier) l'annuaire de l'Internet. À la base, c"est lui qui va permettre de connaitre l'adresse d'une machine ou d'un service à partir de son nom. Un énorme progrès comparé à la maintenance du fichier `/etc/hosts` qu'il fallait, à la vieille époque, télécharger toutes les nuits...

45.1 Outils de base

45.1.1 host

C'est vraiment la commande *vanilla* pour tout ce qui touche à la résolution de nom. Elle propose les premières options de base, avec une syntaxe facile à retenir. Elle existe, sous des formes légèrement déviantes, dans une grande partie des OS [1] existants de nos jours.

```
tth@plop:~ $ host sigfood.dinorama.fr
sigfood.dinorama.fr has address 91.121.221.123
tth@plop:~ $ host -t mx dinorama.fr
dinorama.fr mail is handled by 20 mx1.smeuh.org.
tth@plop:~ $
```

Ces deux exemples sont encourageants. Est-il possible d'aller plus loin ?

45.1.2 nslookup

> Nslookup is a program to query Internet domain name servers. Nslookup has two modes : interactive and non-interactive. Interactive mode allows the user to query name servers for information about various hosts and domains or to print a list of hosts in a domain. Non-interactive mode is used to print just the name and requested information for a host or domain. »

```
tth@tranxene:~$ nslookup -type=txt tetalab.org
```

1. même dans le sale, c'est dire.

```
Server:         80.67.169.12
Address:        80.67.169.12#53

Non-authoritative answer:
tetalab.org     text = "v=spf1 ip4:89.234.156.223 -all"

Authoritative answers can be found from:

tth@tranxene:~$
```

45.1.3 dig

Dig (*domain information groper*) est un outil d'interrogation des serveurs DNS. Par défaut, il questionne les serveurs listés dans `/etc/resolv.conf`.

Pour résoudre récursivement un nom de domaine à partir de la racine : `$ dig +trace <nom.de.domaine>`

Pour obtenir le reverse d'une adresse IP : `$ dig +trace -x <adresse.ip>`

45.1.4 checkresolv

Un outil très basique conçu et bricolé par Tonton Th au siècle dernier lors d'un séjour hivernal à Frontignan-plage. Hélas, les sources semblent définitivement perdues dans les ténèbres de l'Undernet profond. Si quelqu'un a une piste...

Une ré-écriture s'impose, en repartant sur les mêmes bases. En utilisant les mêmes fonctions qu'une application basique. Peut-être aussi en lui trouvant un nom plus *fun*...

45.2 DHCP

> The DHCP protocol allows a host to contact a central server which maintains a list of IP addresses which may be assigned on one or more subnets. A DHCP client may request an address from this pool, and then use it on a temporary basis for communication on network.

Question : comment peut-on « écouter » les requêtes et les réponses DHCP sur notre réseau local pour détecter le rogue-machin ? Probablement avec `tcpdump`, et c'est à l'essai dans la page 61.

Un compagnon privilégié du DHCP est le système PXE qui permet de *booter* des images ISOs stockées sur un serveur du réseau local.

45.3 dnsmasq

> dnsmasq is a lightweight DNS, TFTP and DHCP server. It is intended to provide coupled DNS and DHCP service to a LAN.

Chapitre 46

ssh

Secure Shell.

D'après le site `openssh.org` : *OpenSSH is a* free *version of the SSH connectivity tools that technical users of the Internet rely on. Users of telnet, rlogin, and ftp may not realize that their password is transmitted across the Internet unencrypted, but it is. OpenSSH encrypts all traffic (including passwords) to effectively eliminate eavesdropping, connection hijacking, and other attacks. Additionally, OpenSSH provides secure tunneling capabilities and several authentication methods, and supports all SSH protocol versions.*

46.1 X11 forwarding

Option `-X` et ses implications sur la sécurité.

46.2 sshfs

D'après la manpage Linux : *SSHFS (Secure SHell FileSystem) is a file system for Linux (and other operating systems with a FUSE implementation, such as Mac OS X or FreeBSD) capable of operating on files on a remote computer using just a secure shell login on the remote computer. On the local computer where the SSHFS is mounted, the implementation makes use of the FUSE (Filesystem in Userspace) kernel module. The practical effect of this is that the end user can seamlessly interact with remote files being securely served over SSH just as if they were local files on his/her computer. On the remote computer the SFTP subsystem of SSH is used.*

Hélas, ça ne marche pas trop bien depuis un OpenBSD [1] :

```
~ $ uname -a
OpenBSD vierge.thsf.net 5.5 GENERIC#159 sparc64
~ $ sshfs tth@10.20.0.23:/tvbruits 23/
fuse_mount: Permission denied
```

1. OpenBSD 5.5 sur sparc64

Grace à `<semarie>`[2] : pour un montage en tant que yuser il faut deux choses : les bons droits sur `/dev/fuse*` et un `sysctl kern.usermount=1` bien senti.

La semaine prochaine, nous verrons le *mapping* entre les UID locaux et distants.

46.3 Port forwarding

2. du Groupuscule des Contributeurs d'Unix

Chapitre 47

Web servers

47.1 Apache, Nginx

ìndexnginx

Apache (*A patchy server* et Nginx sont de grands classiques, voire même, dans le cas d'Apache, un grand ancien.

Que nous ne verrons pas pour le moment. Je vais plutôt me concentrer sur un autre serveur : `lighthttp`, pour passer ensuite à quelque chose d'encore plus bas-niveau.

47.2 Lighttpd

Un outsider ?

```
root@ramen:~# lighty-enable-mod userdir
Enabling userdir: ok
Run "service lighttpd force-reload" to enable changes
root@ramen:~# service lighttpd force-reload
root@ramen:~#
```

Q : Sur une installation fraiche, la racine du site est en erreur 403 ? **R :** Par défaut le listing des répertoires vides (sans `index.html`) est interdit. Pour changer ça, il faut `# lighttpd-enable-mod dir-listing` et `# service lighttpd force-reload`, voilà.

Question : comment forcer le `Content-Type` pour un type particulier de fichier ? Par exemple text/plain pour les fichiers `.POV` ?

47.3 Custom made

```
tth@ramen:~/Houba/Fractales90$ apt show libmicrohttpd12
Package: libmicrohttpd12
Version: 0.9.72-2
```

```
Priority: optional
Section: libs
Source: libmicrohttpd
Maintainer: Daniel Baumann <daniel.baumann@progress-linux.org>
Installed-Size: 211 kB
Depends: libc6 (>= 2.17), libgnutls30 (>= 3.7.0)
Homepage: https://www.gnu.org/software/libmicrohttpd/
Tag: role::shared-lib
Download-Size: 109 kB
APT-Sources: http://deb.debian.org/debian bullseye/main amd64 Packages
Description: library embedding HTTP server functionality
 GNU libmicrohttpd is a small C library that is supposed to make it
 easy to run an HTTP server as part of another application.
 Key features that distinguish GNU Libmicrohttpd from other
 projects are:
  * C library: fast and small
  * API is simple, expressive and fully reentrant
  * Implementation is HTTP 1.1 compliant
  * HTTP server can listen on multiple ports
  * Four different threading models (select, poll, pthread, thread pool)
  * Support for IPv6
  * Support for SHOUTcast
  * Support for incremental processing of POST data (optional)
  * Support for basic and digest authentication (optional)
  * Support for SSL3 and TLS
```

Chapitre 48

Hardware

48.1 Joystick

La manette de jeu est un périphérique que j'affectionne particulièrement.

```
int                foo, joy_fd;
struct js_event    js;

joy_fd = open(joy_device , O_RDONLY);
foo = read(joy_fd, &js, sizeof(struct js_event));
```

48.2 Diagnostics

Quand les choses ne se passent pas comme prévu, quand il y a des accrocs dans le plan, que le résultat n'est qu'une video totalement noire de 17 minutes. dmesg, lshw, lsusb, lspci...

48.2.1 lshw

lshw is a small tool to extract detailed information on the hardware configuration of the machine. It can report exact memory configuration, firmware version, mainboard configuration, CPU version and speed, cache configuration, bus speed, etc. on DMI-capable x86 or IA-64 systems and on some PowerPC machines (PowerMac G4 is known to work).

48.3 Ports série

RS-232 (parfois appelée EIA RS-232, EIA 232 ou TIA 232) est une norme standardisant une voie de communication de type série. Le standard RS-232 recouvre plusieurs autres standards : les recommandations UIT-T V.24 (définition des circuits) et V.28 (caractéristiques électriques), ainsi que la norme ISO 2110 pour la connectique.

Comment détecter si un *device* correspond bien à un port série ?

```
#include <stdio.h>
#include <fcntl.h>
#include <termios.h>

#define TTY "/dev/ttyS18"

int main(void) {
        int fd = open(TTY, O_RDWR | O_NOCTTY);
        if (fd < 0) {
                perror("open␣" TTY);
                return 1;
        }

        struct termios tio;
        if (tcgetattr(fd, &tio)) {
                perror("tcgetattr␣" TTY);
                return 1;
        }

return 0;
}
```

Et la question qui, je le sens, va venir : « à quoi peut bien servir un port série ? » appelle un réponse presque évidente : « à plusieurs choses ». Une petite liste : Connecter un Minitel pour dialoguer avec Ulla Trentsixquinze ? Brancher un Sportster 14400 pour retrouver Fidonet ? Discuter avec un équipement avionnique ? Rediriger l'impression d'un MSX vers Cups ?

Les possibilités sont infinies. C'est votre imagination qui est la limite [1]

48.4 Ports //

périphérique `/dev/parport`

Question : comment faire du *bitbanging* ?

48.5 Disques durs

Iozone ? Bonnie ? Smart ? UUID ? Ddrescue ?

48.5.1 blkid & lsblk

À mi-chemin entre le matériel et le système, la commande `blkid` affiche quelques informations sur les périphériques blocs et surtout les partitions qu'ils contiennent :

1. Comme à peu près tout dans la vie.

- Périphérique dans `/dev/`
- UUID XXX
- Type de la partition
- Label textuel
- PARTUID

Quand à `lsblk`, elle affiche le même genre d'information, mais en plus joli (j'ai utilisé l'option `-i` pour que le résultat passe mieux à l'impression) :

```
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0 232.9G  0 disk
+-sda1   8:1    0  18.6G  0 part /
+-sda2   8:2    0   3.8G  0 part [SWAP]
+-sda3   8:3    0   488M  0 part /boot
+-sda4   8:4    0   210G  0 part /home
```

48.5.2 Smart

Self-Monitoring, Analysis and Reporting Technology [2] est un ensemble de choses pour la surveillance de l'état des disques durs et dérivés (SSD). Il permettrait, dans un monde idéal, la prédiction des pannes.

```
smartctl -a /dev/sda
Print a large amount of SMART information for drive /dev/sda .
```

48.5.3 fsck

Check and repair a Linux filesystem

Fortune du jour : Franchement, là, je préfère que ce soit vous qui prenne la tronçonneuse, parce que là, c'est pas beau à voir...

48.5.4 tune2fs

Adjust tunable filesystem parameters on ext2/ext3/ext4 filesystems [3].

48.6 Sensors

L'outil pour connaitre la température de votre CPU surmené.

48.7 Interface réseau

On a parfois besoin de savoir dans quel état est une interface réseau Ethernet (UP, RUNNING dans la sortie de ifconfig). Parfois, je n'aime pas trop *parser*

2. http ://www.smartmontools.org/
3. You can tune a fs, but you can't tuna fish

celle-ci à grand coup de *regexp*, en particulier dans un contexte de type Busybox. Et pour ce cas-là, je préfère utiliser un binaire brassé à la maison, et dont voici un fragment de source :

```
int get_if_flags(char *ifname, short *p_flags)
{
int                     fd;
int                     retcode;
struct  ifreq           req;

#if DEBUG_LEVEL
fprintf(stderr, ">>> %s \"%s\" to %p\n",      \
                    __func__, ifname, p_flags);
#endif

/* Sanity check */
if ( strlen(ifname) > (IFNAMSIZ-1) )
        {
        fprintf(stderr, "name %s to long\n", ifname);
        abort();
        }

fd = socket(PF_INET, SOCK_DGRAM, 0);
if (fd < 0)
        {
        perror("socket bla...");
        return -2;
        }

/* populate the struct for the request */
memset(&req, 0, sizeof(req));
strcpy(req.ifr_name, ifname);

/* do the call */
retcode = ioctl(fd, SIOCGIFFLAGS, &req);
if (retcode < 0)
        {
        perror("ioctl SIOCGIFFLAGS");
        close(fd);
        return -1;
        }
#if DEBUG_LEVEL
/* display the result */
fprintf(stderr, "flags = 0x%04x\n", req.ifr_flags);
#endif

close(fd);

*p_flags = req.ifr_flags;
return 0;
}
```

Hélas, je n'ai pas pu trop tester ce truc sur beaucoup de systèmes, et je vous propose de ne pas trop lui faire confiance pour une application critique...

Chapitre 49

Gadgets

49.1 Fortune

Un rapide petit exemple, avant de passer aux choses techniques :

```
tth@plop:~$ fortune
You will outgrow your usefulness.
tth@plop:~$ fortune
Future looks spotty.  You will spill soup in late evening.
tth@plop:~$ fortune
You are deeply attached to your friends and acquaintances.
tth@plop:~$
```

Bref, vous voyez le principe : c'est un machin à raconter des trucs plus ou moins aléatoires [1]. Il y a tout un tas d'options intéressantes, la principale étant `-o` pour avoir les fortunes offensives qui peuvent choquer les âmes sensibles.

Nous allons donc maintenant le plier à notre volonté en créant nous même le contenu adéquat. Pour cela, nous devons tout d'abord préparer un fichier texte contenant tous nos aphorismes, séparés par une ligne contenant uniquement le caractère `%`.

Ensuite, il faut utiliser `strfile` qui va préparer une version indexée de notre fichier texte afin qu'il soit utilisable par fortune.

Un petit exemple ?

Bah yen a pas...

Le pendant graphique est `xcowsay`.

49.2 Figlet

Display large characters made up of ordinary screen characters.

Il y a énormément d'option, en voici une première : `-W` permet d'espacer un peu les caractères.

1. et jamais revenus, il faut bien le préciser.

```
tth@redlady:~/Devel/TetaTricks$ figlet -W figlet
   __   _           _          _
  / _| (_)   __ _  | |   ___  | |_
 | |_  | |  / _` | | |  / _ \ | __|
 |  _| | | | (_| | | | |  __/ | |_
 |_|   |_|  \__, | |_|  \___|  \__|
            |___/
```

49.3 /usr/games/

On trouvera dans ce répertoire une collections de jeux anciens, dont certains datent d'une époque où la `vt102` était un terminal de luxe. Certains remontent même à l'antique Unix de Berkeley. Et quelques-uns d'entre eux sont devenus, au fil du temps, de grands classiques.

49.3.1 les générateurs

Les décideurs pressés seront heureux de pouvoir enrichir leurs PointPuissant avec de très écoles-anciennes messages cryptiques avec ces quelques outils ergonomiques et pipelinables avec d'autres fidèles serviteurs.

```
tth@plop $ morse -s we make porn | tr '\n' ' ' ; echo
 .--  .    --  .-  -.-  .    .--.  ---  .-.  -.    ...-.-
tth@plop $ ppt we make porn | tr 'o' '*'
-----------
| *** .***|
| **  .* *|
|  *  .   |
| ** *.* *|
| **  .  *|
| ** *. **|
| **  .* *|
|  *  .   |
| *** .   |
| ** *.***|
| *** . * |
| ** *.** |
-----------
tth@plop $ bcd "we make porn" | tr ']' 'x'
 ------------------------------------------------
/WE MAKE PORN                                    |
| x  x x                                         |
|   x x  xxxx                                    |
|x                                               |
|1111x1111111111111111111111111111111111111111111|
|22222x22222222222222222222222222222222222222222|
|33333333333333333333333333333333333333333333333|
|444x444444444444444444444444444444444444444444|
|5x5555x5555x55555555555555555555555555555555555|
|x66666666x66666666666666666666666666666666666666|
|77777777x77777777777777777777777777777777777777|
```

```
|888888888888888888888888888888888888888888888888|
|9999999999x99999999999999999999999999999999999999|
|_________________________________________________|
```

Certains esprits ouverts pensent que `fortune` est aussi un générateur de données futilement complexes. D'autres penseront à MX-Doctor ou Camélia.

49.3.2 Les animations

Ahem, là, c'est délicat, parce que je ne sais pas vraiment [2] faire d'animation imprimable.

49.4 Faire *beep-beep*

Dans le temps, les ordinateurs de type IBM-PC n'avaient qu'un tout petit *buzzer* pour se faire entendre [3] avec seulement 1 bit de résolution. On pouvait quand même modifier la fréquence et jouer de petites mélodies.

```
$ beep -f 1000 -n -f 2000 -n -f 1500
$ beep -f 300.7 -r 2 -d 100 -l 400
$ cat codes.txt | beep -c -f 400 -D 50 -l 10
$ beep -f 1000 -r 2 -n -r 5 -l 10 --new
```

49.5 Les nhorloges

Xclock, Oclock, Swisswatch, Xdaliclock...

2. Enfin, si, j'essaye de confectionner un soft pour fabriquer des flipbooks à partir de portnawak

3. Négligeons les ventilateurs en première approche.

Chapitre 50

Sofware Defined Radio

Beaucoup d'empirisme danc ce chapitre.

Tout d'abord, le matériel à ma disposition :

```
Bus 004 Device 010: ID 0bda:2838 Realtek Semiconductor Corp. RTL2838 DVB-T
```

Ensuite, le premier hack, dont je n'ai pas encore bien capté la raison :

```
$ cat /etc/modprobe.d/blacklist-dvb_usb_rtl28xxu.conf
# added by tth
blacklist dvb_usb_rtl28xxu
```

50.1 CubicSDR

CubicSDR is a cross-platform Software Defined Radio application which allows you to navigate the radio spectrum and demodulate any signals you might discover. It currently includes several common analog demodulation schemes such as AM and FM and will support digital modes in the future. SDR hardware is accessed through SoapySDR and therefore any hardware for which a SoapySDR module exists can be used as a receiver with CubicSDR.

Chapitre 51

Bases de données

51.1 sqlite

51.1.1 CLI

Commandes internes :

51.1.2 Libreoffice

51.2 GDBM

> GDBM - The GNU database manager. Includes dbm and ndbm compatibility.

Gnu DBM permet de stocker des paires clef/valeur dans un fichier de données. Chacune de ces clefs doit être unique, et doit être liée avec une seule valeur [1].

51.2.1 Exemple en Perl

```
#!/usr/bin/perl  -w

use strict;
use GDBM_File;
my $nomdb = "exemple.gdbm";
my (%DB, @champs);
my ($user, $gecos);
open(SOURCE, "< /etc/passwd") or die "source $!";
tie(%DB, "GDBM_File", $nomdb, GDBM_WRCREAT, 0666) or die "gdbm $!";

while (<SOURCE>) {
    @champs = split ":", $_;
    $user = $champs[0]."\0";
```

1. Peut-on alors parler de bijection relationnelle ?

```
    $gecos = $champs[4]."\0";
    $DB{$user} = $gecos;
    }

untie %DB;
close SOURCE;
0;
```

Vous remarquerez que pour cet exemple, nous avons clef et valeur qui sont du texte. Alors que DBM est prévu pour stocker des blocs de binaire. Pour faciliter la récupération de ces chaines de caractères par un hypothétique [2] programme en C, le programme Perl rajoute un `'\0'` (fin de chaine) aux deux champs concernés.

51.3 Postgresql

Le poids lourd du domaine.

51.4 Et le "NoSQL" ?

2. Oui, je vais l'écrire bientôt.

Chapitre 52

S-lang

52.1 C'est quoi ?

S-lang est une bibliothèque de fonctions multi-plateforme pour faire des trucs sympathiques dans des consoles texte. L'exemple le plus éblouissant étant le célèbre `slrn` : "s-lang news reader", dont on cause plus longuement dans les pages consacrées à Usenet (page 144).

52.2 slsh

Autour de s-lang, on trouve un interpréteur de scripts.

Voyons un premier exemple d'utilisation, inspiré de l'exemple de la man-page [1], et qui montre l'utilisation en mode interactif :

```
tth@plop:~/Essais/Slang$ slsh
slsh version 0.8.4-1; S-Lang version: 2.2.2

slsh> 3*10;
30
slsh> x = [1:20];
slsh> sum (sin(x)-cos(x));
0.45861331748677714
slsh> print x;
Usage: print (OBJ [,&str|File_Type|Filename]);
Qualifiers: pager[=pgm], nopager
slsh> ^D
tth@plop:~/Essais/Slang$
```

Cool...

1. vous lisez bien les pages de man ?

Chapitre 53

curses

En fait, nous allons plutôt parler de **ncurses** qui est l'implémentation actuelle de l'ancêtre `curses`. Voici ce qu'en dit une page de man :

> The ncurses library routines give the user a terminal-independent method of updating character screens with reasonable optimization. This implementation is "new curses" (ncurses) and is the approved replacement for 4.4BSD classic curses, which ha been discontinued.

Plus concrètement, Curses est un moyen optimisé de gestion des écrans textes (Rappellez-vous notre ami le MINITEL) afin de créer des interfaces 'full-screen'. Vous pouvez adresser chaque cellule de votre écran (la case où l'on peut mettre un caractère) pour y écrire quelque chose, en gérant des attributs tels que gras ou souligné.

53.1 Les principes

- Indépendance de l'écran physique
- Fenêtre et sous-fenêtre
- Deux écrans : stdscr et curscr

Au tout début était le terminal canonique, la TELETYPE. C'est dans le doux cliquetis de la vénérable `ASR33` que se sont échaffaudées les premières lignes du code, écrites avec *the only true one*[1]. Ensuite sont arrivés les terminaux à écran vidéo, les fameuses *Glass TTYs*, qui ne tardèrent pas à se munir d'un curseur adressable, ce qui permet alors d'écrire un caractère à n'importe quelle endroit de l'écran.

Adm3a, vt52, Qume 60...

Une structure essentielle dans curses est l'écran, appelé *screen* dans l'original. C'est une matrice rectangulaire de cellules caractère/attribut et quelques données associées.

Pourquoi deux écrans ? La mécanique interne de curses maintient en interne deux écrans (qui sont en fait des fenêtres de la taille de l'écran physique) connus

1. ed

sous les noms de `stdscr` et `curscr`. Le premier est celui que voit l'application, et qu'il peut manipuler avec les fonctions (genre `mvaddstr`) que nous verrons très bientôt. Et le second est l'image interne que se fait la mécanique de l'écran physique, donc ce que voit le yuser.

53.2 Premier exemple

Bien entendu, nous commencerons par l'ECM traditionnel de rigueur.

```
#include <stdio.h>
#include <stdlib.h>
#include <ncurses.h>
int main(int argc, char *argv[])
{
int             key;

if (2 != argc) {
        fputs("need␣an␣argument\n", stderr);
        exit(1);
        }
initscr();              /* first initialization */
cbreak();               /* no line buffering */
noecho();               /* be silent on input */
keypad(stdscr, TRUE);   /* acces touches curseur */

mvaddstr(10, 3, argv[1]);
refresh();
key = getch();
endwin();
printf("code␣touche␣␣%d␣␣0x%X\n", key, key);

return 0;
}
```

La première étape consiste à initialiser et configurer le moteur interne de ncurses, et à régler l'interaction clavier/écran.

Ensuite, avec `mvaddstr`, nous écrivons le texte passé en paramètre à une position fixée ligne,colonne dans un écran virtuel (qui n'existe qu'en mémoire) qui sera ensuite transféré dans l'écran réel (celui qui est affiché) par le `refresh()`.

Dans ma grande mansuétude, je n'aurais pas l'audace de vous montrer une capture d'écran, puisque je me permet de supposer que vous avez pris le temps de compiler l'exemple, et de voir par vous même ce que ça donne.

53.3 Bell and whistles

53.3.1 Attributs

Voici comment centrer sur la ligne du haut un texte écrit en **gras** grace à l'attribut `A_BOLD` :

```
attron(A_BOLD);
mvaddstr(0, (COLS / 2) - (strlen(text) / 2), text);
attroff(A_BOLD);
```

Il existe tout un tas d'autres attributs : `A_UNDERLINE` pour le soulignement ou `A_BLINK` pour clignoter, par exemple. La liste complète est dans le man : `attr(3NCURSES)`, avec plein de fonctions spécialisées dans le traitement des attributs.

53.4 Les fenêtres

53.5 Astuces

Dans une boucle interactive, on va utiliser `getch()`, qui va bloquer tant qu'on n'a pas tapé sur une touche. Mais que faire si on veut utiliser, en plus du clavier, un autre périphérique de saisie, genre un Joystick ?

Bien entendu, on peut utiliser `select(2)`, mais cela nécessite de connaitre le *file descriptor* auquel est attaché le clavier. Dans le contexte classique, on peut assumer que ce sera le `fd` associé à *stdin*, mais il est imprudent de compter là-dessus.

—> voir `newterm(3)`

Chapitre 54

WiFi

WIreless FIdelity.

54.1 wicd

Wired and Wireless Network Connection Manager.

Wicd est constitué de deux éléments : un *daemon* qui est responsable de gérer les connections, et plusieurs interfaces utilisateurs. Ces deux parties communiquent par D-Bus.

```
root@plop:~# wicd-cli -S -y
root@plop:~# wicd-cli -l -y
#       BSSID                   Channel ESSID
0       00:25:15:A6:2A:D0       1       NEUF_2ACC
1       CA:25:15:A6:2A:D3       1       SFR WiFi Mobile
2       CA:25:15:A6:2A:D1       1       SFR WiFi Public
3       A0:21:B7:D5:BA:DB       6       NUMERICABLE-35E0
4       00:17:33:D4:39:B8       11      NEUF_39B4
5       F4:CA:E5:95:DA:1E       10      FreeWifi_secure
6       F4:CA:E5:95:DA:1D       10      FreeWifi
7       16:10:18:47:F2:4C       11      <hidden>
8       00:24:D4:CE:4F:A1       1       FreeWifi
9       00:24:D4:CE:4F:A2       1       FreeWifi_secure
10      16:10:18:47:F2:4E       11      FreeWifi_secure
11      00:24:D4:CE:4F:A0       1       freeboxwifi
12      00:0E:9B:24:F5:2B       7       poisson_bulle
```

La première ligne demande au *daemon* de scanner (option `-S` ou `-scan`) les réseaux disponibles en wireless. Puis la seconde commande demande l'affichage de la liste.

54.2 Un coup de NAT

Une machine connectée en wifi sur une friboite [1] peut facilement faire translateur d'adresse pour un réseau branché sur son eth, à condition de connaitre les incantations magiques (qui sont ici très linuxisantes) :

```
ifconfig eth0 10.9.9.200/24
echo 1 > /proc/sys/net/ipv4/ip_forward
modprobe iptable_nat
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

Ensuite sur les machines du lan, donc le coté ethernet, il faut incanter ce genre de trucs :

```
ifconfig eth0 10.9.9.199/24
route add default gw 10.9.9.200
echo "nameserver 192.168.0.254" >> /etc/resolv.conf
```

Normalement, ça devrait marcher. D'un autre coté, ce texte date de 2011, et l'arrivée de `systemd` a peut-être cassée des choses. Et c'est ce que nous allons voir dès maintenant...

9 juillet 2021 : C'est tout vu, ça ne fonctionne pas, et je pense que c'est la rêgle iptables qui n'est plus d'actualité, ou le paramêtre `wlan0` qui a (encore ?) changé de nom.

54.3 Le NAT avec `nmcli`

Nmcli est un outil en ligne de commande pour contrôler NetworkManager. Il a été prévu pour être scriptable.

1. ou le Cisco sauvage de la salle d'AG, hein, mais ça ne sera pas le même nameserver.

Chapitre 55

Trucs divers

55.1 Permissions

Suite à un accident d'origine indéterminé, Captain ROOT a crée un fichier dans le $HOME d'un pauvre péone. Et celui-ci découvre avec effroi qu'il ne peut ni lire ni modifier ce fichier. Ce que peut confirmer un `ls -l` qui montre bien que seul root a des droits sur ce fichier :

```
-rwx------  1 root  root    70 2012-03-15 12:16 coin.pan
```

Question : qui tombe à l'eau ?

55.2 Prompt

```
printf '\033]11;lemonchiffon2\a'
```

Pour être honnete, je n'ai pas la moindre idée de ce que cela veut dire. Mais j'imagine que le `*a` à la fin est destiné à activer la cloche.

55.3 fdupes

Un utilitaire pour découvrir les fichiers dupliqués dans votre disque dur qui se remplit trop vite à votre gout.

55.4 g77 : Command not found

Lors d'une fin de soirée, je me suis retrouvé à fouiller dans ma portion du Ternet, et je suis tombé sur un très vieux truc. Des fractales en FORTRAN77, avec des incantations comme celle-ci dans le Makefile :

```
map_henon: map_henon.f
g77 $(G77OPT) map_henon.f -o map_henon -limage
```

Hélas, je n'ai pas de commande `g77` dans ma machine, que faire ? En écrire une ?

55.5 qemu

http ://sven.stormbind.net/blog/posts/deb_qemu_local_openwrt/

Chapitre 56

On est bientôt arrivé ?

Non, la route est longue, mais la voie est libre.

Si vous êtes arrivés jusque ici, c'est que vous avez (peut-être ?) parcouru cet ouvrage, et constaté à quel point il est en friche. Beaucoup de chapitres sont vraiment incomplets, voire cruellement vides, ou même obsolètes à un niveau pitoyable.

Mais les choses vont évoluer, peut-être, un jour, on ne sait pas quand...

Index