Table des Matières

1.Présentation **3**
1.1.Introduction 3
1.1.1.Travaille demandé 3
1.1.2.Approche du problème 3
1.1.3.Moyen mis en oeuvre 3
1.2.La DreamCast et Linux 4
1.2.1.Processeur central 4
1.2.2.Le lecteur de GD-ROM 4
1.2.3.Le processeur graphique 4
1.2.4.Le modem 4
1.2.5.Les périphériques d'entrée 5
1.3.Séquelles du T.E.R 5
1.3.1.The good... 5
1.3.2. ...the bad... 5
1.3.3. ... and the ugly 5

2.Js2Mouse **6**
2.1.Introduction 6
2.1.1.Pourquoi ? 6
2.1.2.Protocole souris PS2 6
2.1.3.Événements 7
2.2.Traduction des touches 7
2.2.1.Création de la carte de traduction 7
2.2.2.Traduction en exécution 8
2.3.Conversion 8
2.3.1.Fonctionnement 8
2.3.2.Mouvements 9
2.3.3.Clics 10
2.4.Évolutions 11
2.4.1.Création de la carte de traduction 11
2.4.2.simplification du code 11
2.4.3.Boutons analogiques 11
2.4.4.Passage dans l'espace noyau 11

3.Dream[Slack] **12**
3.1.La base 12
3.1.1.Tutorial de Bill Gatliff 12
3.1.2.DCLinux 12
3.1.3.Dream[Slack] 13
3.1.4.Implémentation de Dream[Slack] 14
3.2.Logiciels clés 14
3.2.1.Links 2 14
3.2.2.Boa 16
3.3.Applications 16
3.3.1.Exemples 16
3.3.2.Implémentation de l'interface 18
3.3.3.Idées et évolution 21

4.Annexes **22**
4.1.Références 22
4.2.Bibliographie 22
4.3.Étude de DCLinux (en Anglais) 23
4.3.1.ECos as boot loader 23
4.3.2.System on the ramdisk 23
4.3.3.System on the gdrom 23
4.3.4.Result 25
4.4.Construction de Dream[Slack] 25

1. Présentation

1.1. Introduction

1.1.1. Travaille demandé

Le but de ce Travail d'Étude et de Recherche (T.E.R) est la création d'une distribution Linux [1], nommée "Dream[Slack]", pour la console de jeu SEGA DreamCast [2]. Tout le monde, et surtout les non-informaticiens, doivent pouvoir l'utiliser sans ajout d'adaptateurs clavier, souris ou VGA.

Le choix de cette machine a été déterminé par rapport au modem intégré, son très faible coup, ainsi que par la facilité à pouvoir développer librement dessus.

1.1.2. Approche du problème

Pour que n'importe quel utilisateur 'traditionnel' de DreamCast puisse utilisé Linux sur cette machine, il faut qu'il soit capable de le faire depuis son joystick. Ce T.E.R. se divise donc en deux parties.

La première partie consiste en l'écriture d'un pilote qui simule une souris à partir des entrées du joytsick. En effet, beaucoup plus de logiciels utilisent la souris que le joystick.

La seconde partie est la distribution Dream[Slack] en elle-même. On y retrouve un serveur HTTP qui fournis des pages permettant de lancer des applications ou de configurer le système, cela grâce des scripts s'exécutant sur le serveur. Afin de combler le manque de claviers, les pages HTML sont étendu par l'ajout de JavaScript, ce qui permet de créer un clavier virtuel accessible depuis la souris, donc depuis le joystick.

1.1.3. Moyen mis en oeuvre

Pour pouvoir développer librement sur DreamCast, il faut posséder, soit un câble de liaison entre le port série du PC et celui de la DreamCast., soit un adaptateur Ethernet. Il faut aussi utilisé la compilation croisée, c'est à dire que les programmes sont compilés sur une architecture différente de celle de la 'cible', ensuite, ils sont injectés dans la DreamCast grâce aux adaptateurs sus-mentionnés.

Tout cela fait énormément penser au développement sur systèmes embarqués, et les caractéristiques de la DreamCast vont dans ce sens : processeur SH4, 16 Mo de mémoire RAM, pas de disque dur.

1.2. La DreamCast et Linux

1.2.1. Processeur central

Le processeur central est un Hitachi SH4 Little Endian cadencé à 200 MHz et munie d'un FPU. C'est un processeur 128 bits, mais qui peut travailler en 16, 32, 64 bits selon les opérations. Les membres de LinuxSH [3] sont les mainteneurs du noyau sur les processeurs SH, ils s'occupaient aussi de l'écriture de pilotes de périphériques pour les machines basées sur cette architecture, y compris la DreamCast. Aujourd'hui, ce sont les développeurs de LinuxDC [4] qui ont pris le relaie pour cette machine.

1.2.2. Le lecteur de GD-ROM

La DreamCast ne possède pas un lecteur de CD-ROM traditionnel, mais un lecteur capable de lire un format différent : le Gigabytes Disc-ROM. Celui-ci permet de stocker jusqu'à 1 Go de données sur un support de 12 Cm de diamètre. Au niveau de la vitesse de lecture, le lecteur de GD-ROM tourne à 12x ce qui donne un flux de 1800 KB/s.

Je ne sais pas si Linux peut lire les GD-ROM, de plus seules les compagnies autorisées par SEGA peuvent graver ce genre de disque. En revanche, la lecture de CD-ROM est fonctionnelle, ce qui permet d'avoir une système racine en lecture seule de grande capacité, malgré le fait que les accès soit lent par rapport à un disque dur, du NFS, ou à un RamDisk.

1.2.3. Le processeur graphique

Le processeur graphique est un NEC/VideoLogic PowerVR2. Sous Linux, son support est géré par un pilote du FrameBuffer. Cela permet d'avoir une console sur l'écran, et de faire tourner le serveur XFree86-FBDev [5].

Il semblerait que le serveur XFree86 4.2.0 possède un pilote spécifique pour cette carte sur architecture DreamCast, mais je n'ai pas pris le temps de le tester.

1.2.4. Le modem

Fourni d'origine, le modem est un 33,6 kb/s en Europe. Il existe un pilote mais sur un autre système d'exploitation nommé Kallisti OS [6]. Certains pilotes de Linux pour DreamCast ont été tirés des travaux réalisé par les développeurs d'autres systèmes, il ne reste plus réaliser un portage.

1.2.5. Les périphériques d'entrée

Les périphériques d'entrés sont connectés au Maple Bus, une sorte d'USB. Linux pour DreamCast le supporte ainsi que de nombreux périphériques dont le joystick, la souris, le clavier.

1.3. Séquelles du T.E.R

1.3.1. The good...

Comme nous l'avons vu précédemment, ce T.E.R. s'approche beaucoup du développement pour système embarqué. C'est une chose que l'on n'a malheureusement pas abordé pendant le cursus de Licence, et c'est donc avec joie que ce travail m'a permis de découvrir ce domaine. De même, la conception d'une distribution Linux, apporte énormément d'expérience pratique dans le monde des systèmes d'exploitation.

L'Anglais n'étant plus enseigné en second cycle, la lecture de nombreuses documentations ainsi que l'écriture de quelques textes dans la langue de Shakespeare m'ont permis de conserver un niveau technique acceptable.

1.3.2. ...the bad...

Malheureusement, le temps m'ayant fait défaut, trop de choses ne sont pas encore finis, et l'un de mes principaux regrets, est le fait de ne pas avoir pu rédiger ce rapport en Anglais. Ce T.E.R. ne s'arrêtera pas immédiatement, et j'aurais tout le loisir d'implémenter tout ce qui manque. De plus Jalv et Kurdy m'ont proposé de participer à l'élaboration de Dream[Slack] après la présentation au jury, ce qui permettra d'accélérer son développement.

1.3.3. ... and the ugly.

Un problème technique ayant survenue lors des derniers préparatifs de la sortie de la distribution Linux pour DreamCast, je n'ai pas pu fournir une version de démonstration pour la présentation oral. En fait, je n'ai pas su faire fonctionner correctement GPM [7], le logiciel s'occupant de la souris en mode console.

Un petit mal-fonctionnement de Js2Mouse a aussi été découvert tardivement, mais cela n'empêche absolument pas de l'utiliser sans problèmes avec XFree86 ou GPM. Peut-être que d'autres logiciels risquent de ne pas le supporter, mais dans peu de temps un correctif sera appliqué.

2. Js2Mouse

2.1. Introduction

2.1.1. Pourquoi ?

Des convertisseurs d'événements joystick en événements souris existent déjà, malheureusement ils sont tous, à ma connaissance, intégrés à des logiciels tels que GPM, XFree86, ... Js2Mouse est donc un pilote, développer depuis rien, qui a pour unique tâche de réaliser cette conversion. De cette manière, un logiciel nécessitant directement les sorties souris n'a pas besoin d'intégrer dans son code un tel émulateur pour utiliser le joystick.

Le but final de Js2Mouse est de passé dans l'espace noyau, c'est l'une des raisons pour laquelle il est en C. Ainsi, ce pilote gérera son propre tampon et sera capable d'émuler d'autre type de souris, notamment celles qui nécessitent une initialisation à travers le fichier de périphérique ou un appel à *ioctl()*.

2.1.2. Protocole souris PS2

Pour connaître le protocole de ma souris, j'ai utilisé une technique d'analyse inversée. Pour cela MouseTest, fourni avec Js2Mouse, lit le fichier de périphérique du pilote d'une souris et affiche le résultat à l'écran. A partir de là on peut retrouver comment fonctionne le pilote de la souris et l'imiter.

Exemple d'exécution de MouseTest :

```
./mousetest
b[0]= -3 b[1]=  5 b[2]=  0
b[0]= 24 b[1]= -4 b[2]=  3
b[0]=  0 b[1]= 24 b[2]= -5
b[0]=  1 b[1]=  0 b[2]= 24
b[0]= -5 b[1]=  1 b[2]=  0
b[0]= 24 b[1]= -5 b[2]=  0
b[0]=  0 b[1]= 56 b[2]= -5
b[0]= -1 b[1]=  0 b[2]= 56
b[0]= -5 b[1]= -3 b[2]=  0
b[0]= 56 b[1]= -4 b[2]= -6
b[0]=  0 b[1]= 56 b[2]= -2
b[0]= -8 b[1]=  0 b[2]= 56
b[0]= -1 b[1]=-11 b[2]=  0
b[0]= 40 b[1]=  0 b[2]=-11
b[0]=  0 b[1]= 40 b[2]=  1
b[0]=-10 b[1]=  0 b[2]= 40
b[0]=  3 b[1]= -7 b[2]=  0
b[0]= 40 b[1]=  2 b[2]= -4
b[0]=  0 b[1]= 40 b[2]=  3
```

Dans le cas d'une souris PS2, cela a permis de conclure que les événements sont fournis sous la forme d'un tableau de 3 octets. Le premier est un masque binaire représentant les actions en cours, le second représente la vitesse horizontale et le dernier la vitesse vertical.

Extrait du fichier *convert_ps2.c* des sources de Js2Mouse :

```
//The fisrt byte represente ORed actions :

#define PS2_CLICK_LEFT          0x9     //1001
#define PS2_CLICK_RIGHT         0xa     //1010
#define PS2_CLICK_RELEASE       0x8     //1000

#define PS2_MOVE_LEFT           0x18    //1 1000
#define PS2_MOVE_RIGHT          0x8     //1000
#define PS2_MOVE_UP             0x8     /1000
#define PS2_MOVE_DOWN           0x28    //10 1000
```

2.1.3. Événements

Le pilotes de souris renvoie en continue un statut lorsque celle-ci est en mouvement, alors que le pilote de joystick renvoie un événement. Pour être plus clair, tant que la souris ce déplace, le pilote indique sa vitesse et sa direction, alors que lorsque l'axe du joystick est actif mais qu'il ne bouge plus, par exemple en font de buté, le pilote n'envoie pas d'événements.

Le pilote de souris fournis en simultané les actions en cours, alors que pour le joystick, le pilote n'indique que séquentiellement les événements. Ces points sont importants pour la compréhension des sources de Js2Mouse.

2.2. Traduction des touches

2.2.1. Création de la carte de traduction

Lorsque cela sera complètement implémenté, l'utilisateur pourra choisir quelle touche du joystick représente telle touche de la souris, cela s'effectuera par passage de paramètres. Cette fonctionnalité n'est pas encore présente car inutile pour la phase de développement, de plus il faut vérifier que l'utilisateur n'entre pas une carte erronée, ce qui peut être lourd et peu intéressant.

Malgré le fait que cela ne soit pas utilisable pour le moment, une carte par défaut est créer 'en dure' par *j2m_create_default_map()*, pour que le pilote puisse déjà utiliser les fonctions de traduction de touches.

Extrait de *j2m_create_default_map()* :

```
map[0] = number_of_buttons;
map[1] = JS_CLICK_LEFT;
map[2] = JS_CLICK_RIGHT;
map[3] = JS_CLICK_MIDDLE;
map[4] = JS_WHEEL_UP;
map[5] = JS_WHEEL_DOWN;

map[map[0]+1] = JS_MOVE_HORIZONTAL;
map[map[0]+2] = JS_MOVE_VERTICAL;
```

2.2.2. Traduction en exécution

Comme on peut le constater, *map[0]* a un but particulier, il permet de savoir où ce trouve la traduction des événements dues aux axes du joystick, c'est-à-dire après celles des boutons.

Tout d'abord *j2m_map_it()* regarde le type de l'événement, ensuite s'il est du à un bouton, la traduction est recherchée dans le tableau *map* à l'emplacement *joy_event->number+1*, et s'il est du aux axes, elle est recherchée à l'emplacement *map[0]+joy_event->number+1*.

2.3. Conversion

2.3.1. Fonctionnement

Afin de fournir continuellement des événements de la souris virtuelle lors d'un mouvement, il faut que la lecture du joystick soit non bloquante et que la fonction de conversion *j2m_convert()* continue sa simulation selon le statut du joystick. Pour connaître ce dernier, cette fonction renvoie un drapeau : ACTIVED pour indiquer qu'une touche ou un axe a été activé, DESACTIVED lorsque l'un d'eux a été désactivé, NONE lorsqu'un événement redondant est apparu ou qu'une touche non cartographiée a été enfoncée. Ensuite la fonction appelante *j2m_loop()* met à jour le masque binaire *state* et écrit dans le tube si besoin est.

Extrait de *j2m_loop()* :

```
[...]
ret = poll( poll_fd, 1, 20)) //lecture non bloquante mais qui dort pendant 20µs si rien ne se passe.
[...]
ret = j2m_convert( type, joy_event.value, mouse_buff, state);
[...]
switch ( ret ) {
        case NONE :
                if ( !(state & ~((1<< JS_CLICK_LEFT) | (1<< JS_CLICK_RIGHT))) )
                        continue;
        case DESACTIVED :
                if ( 0 == state)
                        continue;
                state &= ~(1<< type);
                break;
        case ACTIVED :
                state |= (1<< type);
                break;
        default :
                break;
}
```

Enfin, *j2m_convert()* réutilisera ce masque pour continuer à simuler le mouvement de la souris et cela même si l'événement en cours est, par exemple, un clic.

Extrait de *j2m_convert()* :

```
#define AXES_ACTIVE ((state & (1<<JS_MOVE_HORIZONTAL)) || (state & (1<<JS_MOVE_VERTICAL)) )

[...]

if ( AXES_ACTIVE || move)
        j2m_accel(mouse_buff); //fonction de simulation des mouvements
```

2.3.2. Mouvements

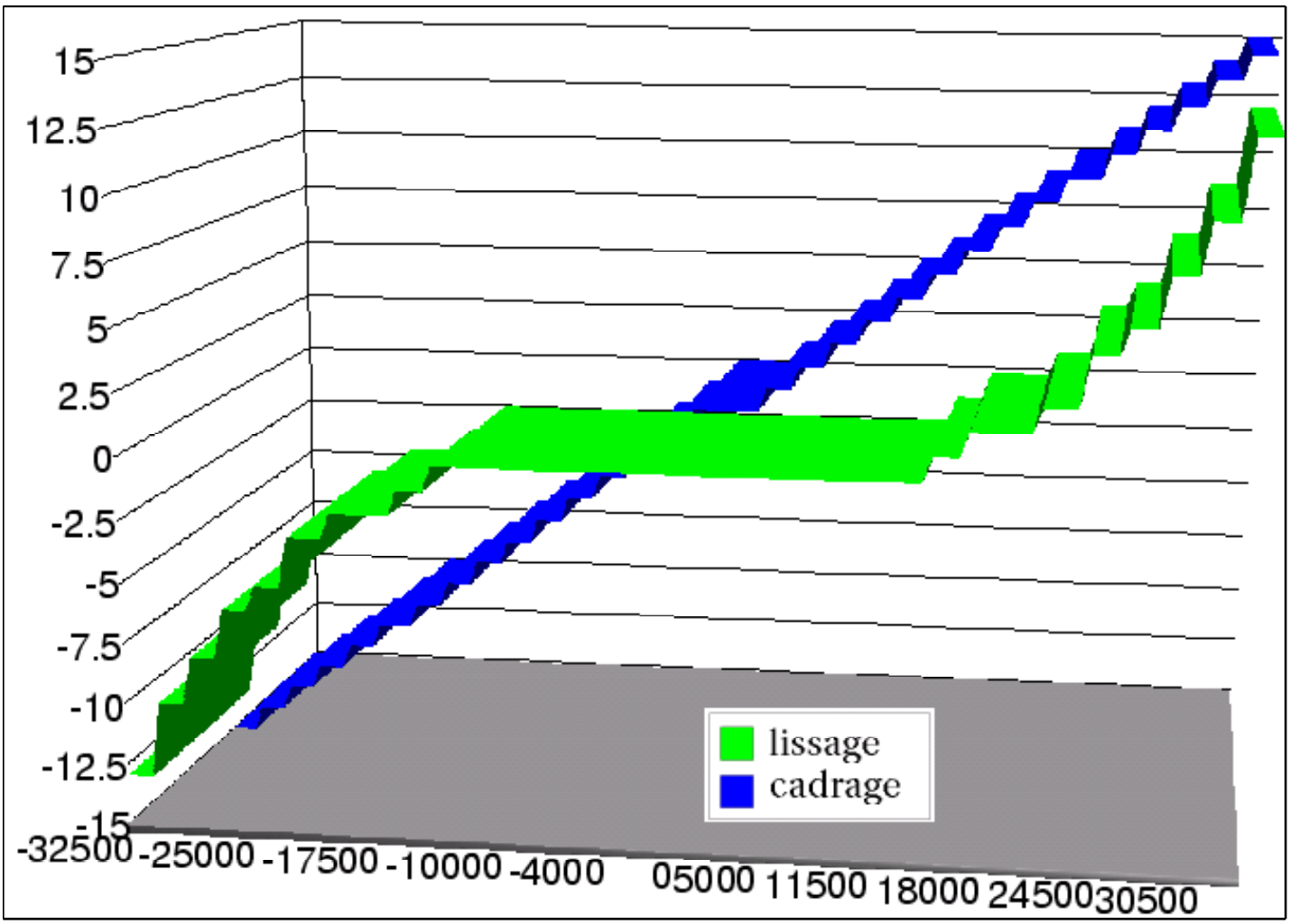
Les valeurs d'inclinaison d'un axe du joystick vont de -32767 à 32767, Js2Mouse recadre ces valeurs entre -16 et 16 pour être acceptable par les logiciels utilisant la souris. Un phénomène étrange apparaît à cause des multiples transtypages, les valeur sont au final comprises entre -13 et 13, mais cela n'est pas gênant.

Ce valeurs ne sont pas converties tels quels, mais sont lissées en étant mises sous forme cubique pour éviter une trop grande sensibilité.

Extrait de j2m_convert() :

```
value = (__s16) (value/2048);
value = (__s16) (value*value*value/256);
```

Graphe des valeurs cadrées puis lissées :



Je ne suis pas pleinement satisfait du résultat, car on obtient une valeur nulle sur une trop grand plage, l'utilisation d'une table de correspondance pourrait être mieux adapté.

L'accélération est gérée de manière très rudimentaire car ce sont les logiciels telle que GPM ou XFree86 qui doivent s'en occuper. Js2Mouse la lisse tout de même pour apporter un peu de confort, mais surtout pour que les joysticks digitaux puissent fonctionner correctement. En effet, avec ces derniers les valeurs d'inclinaison passent de 0 à +/- 32767 sans étapes intermédiaires. Dans notre cas, l'accélération est une fonction constante (+/- 1), donc la vitesse évolue linéairement. C'est la fonction *j2m_accel()* qui s'en occupe.

2.3.3. Clics

En ce qui concerne la simulation des clics de souris, il n'y a rien de particulier à dire.

2.4. Évolutions

2.4.1. Création de la carte de traduction

Cette fonctionnalité est presque implémentée, ce n'est pas ce qui est le plus urgent. En effet, il existe un risque de 'segmentation fault' dans *j2m_create_default_map()*, qui à été découvert pendant la rédaction de ce rapport. Cela peut se produire uniquement si le nombre de boutons du joystick est inférieur à 3.

2.4.2. simplification du code

Le fait que ce soit deux fonctions qui gère la variable *state* rend le code peu lisible. Dans peu de temps, ce sera exclusivement *j2m_convert()* qui s'en occupera.

2.4.3. Boutons analogiques

Sur certains joysticks il existe des boutons analogiques qui sont vus par les pilotes comme des axes. L'ajout de cette fonctionnalité semble indispensable pour que Js2Mouse puisse fonctionner avec un maximum de modèles.

2.4.4. Passage dans l'espace noyau

Extrait de *joystick.txt* des sources du noyau Linux :

> *For all joystick drivers to function, you'll need the userland interface module in kernel, either loaded or compiled in:*
> *modprobe joydev*

C'est donc dans le fichier *joydev.c* que Js2Mouse sera implémenté, en effet c'est par ici que les événements des joysticks arrivent avant de passer dans l'espace d'adressage utilisateur. Il suffira donc de convertir les donnée avant de les envoyer.
Comme la fonction *joydev_write()* n'est pas utilisée, une écriture sur le fichier de périphérique pourrait demander la commutation entre le mode souris et le mode joystick, ou entre différents types de souris.

L'objectif final de ce pilote étant le passage dans le mode noyau, les fonctions principales, disponible dans *map.c* et dans *convert_ps2.c* des sources de Js2Mouse, ont été codées de manière à n'utiliser aucunes fonctions du mode utilisateur. Leurs intégrations en sera facilitées.

3. Dream[Slack]

3.1. La base

3.1.1. Tutorial de Bill Gatliff

Bill Gatliff [8] est un consultant et un spécialiste pour systèmes embarqués, il a écrit de nombreux articles, dont un sur la création d'un système Linux minimal en RamDisk pour la DreamCast. Ce tutorial est une très bonne introduction à la compilation croisée, ainsi qu'aux systèmes embarqués, et pour le moment Dream[Slack] est basée sur ce dernier.

Base de la mini-distribution de Bill Gatliff :

binutils-2.11.2
gcc-3.0.1
glibc-2.2.4
busybox-0.60.1
linux-sh-dc 2.4.10-pre6

3.1.2. DCLinux

DCLinux [9] est la plus connue et la plus aboutie des distributions Linux pour DreamCast, et c'est pour cela qu'elle a énormément servie pour les testes des logiciels que l'on retrouve dans Dream[Slack]. En effet, elle fournissait un environnement très complet et facile à utiliser, on y retrouve notamment une console sur le port série, le support du réseaux, le serveur XFree86, et beaucoup d'autres logiciels. Malheureusement, elle n'est plus maintenue, et les documentations techniques ne sont pas abondantes.

Il s'avère que la dernière version de DCLinux, datée du 29 Septembre 2001, n'a jamais été officiellement distribuée. Contrairement à la précédente, elle n'utilise pas le système DevFS, ce qui empêche la création où la modification des fichiers de périphériques. Cela est extrêmement gênant car lors d'une connexion distante ou locale, ces fichiers spéciaux se voient créer ou changer de propriétaires. C'est pour cela qu'elle ne fut pas utilisée lors des testes. On y retrouve tout de même quelques bonnes surprises comme un noyau plus récent, enfin la librairie pthread, et la librairie DirectFB.

3.1.3. Dream[Slack]

Même si Dream[Slack] est basée sur la mini-distribution de Bill Gatliff, elle est capable de charger des binaires dynamiquement liés, pour cela il suffit de copier *ld-2.2.4.so* dans le répertoire */lib/* et d'exporté la variable *LD_LIBRARY_PATH* ou de mettre à jour le fichier */etc/ld.so.conf* et de lancer *ldconfig*.

Ajout de Dream[Slack] par rapport à la mini-distribution de Bill Gatliff :

linux-sh-dc 2.4.10-pre11
ncurses-5.3
gpm-1.19.6 ou gpm-1.20.0
zlib-1.1.4
libpng-1.2.5
libjpeg.v6b
libtiff-3.6.0-beta
links-2.1pre7
boa-0.94.13

Ces librairies sont compilées en tant que librairies dynamiques, et statiques. Le fait de liés statiquement un binaire permet de gagner du temps sur son chargement, en effet le lecteur de GD-ROM n'a plus besoin de chercher à plusieurs endroits les différentes librairies utilisées. Cependant cela peut être très consommateur en mémoire car les librairies ne seront pas partagées.

D'autres toolchains, fournis par Sugioka [10], ont été testés avec succès, et seront sûrement utilisés pour les prochaines versions de Dream[Slack].

Toolchains fournit par Sugioka :

binutils-2.13.90.0.18
gcc-3.2.3
glibc-2.2.5

Cependant, il semblerait que les dernières version de *binutils*, *gcc*, et de la *glibc* supporte nativement la cible *sh-linux*, si telle est le cas, Dream[Slack] sera basé sur ceux-ci.

3.1.4. Implémentation de Dream[Slack]

Beaucoup de logiciels libres pour DreamCast supportent le changement de CD-ROM en cours d'exécution, pour que l'utilisateur puisse accéder à ces fichiers (de musique par exemple). Malheureusement DCLinux utilise le CD-ROM comme système racine, si Dream[Slack] est construite de cette manière, ce qui est le cas pour le moment, il faudrait que l'utilisateur graves ses fichiers sur ce même CD-ROM.

Une autre solution consisterait à faire une distribution entièrement sur un RamDisk, ce qui signifie qu'il faille faire extrêmement attention à la mémoire utilisée. Il existe des techniques pour ce genre de situation, telle que le stockage des binaires sur le RamDisk sous forme directement exécutable, il n'y a donc plus redondance des données. On pourrait sinon créer des distributions Dream[Slack] dédiées, une pour la lecture de musique, une autre pour l'émulation de jeu, ...

La meilleur solution serait de créer un système capable de mixer les deux techniques, un RamDisk qui possède des liens vers des répertoires du CD-ROM. Ainsi quoi qu'il arrive, il existe toujours un système racine, et si l'utilisateur veut lancer une application disponible sur le CD-ROM de Dream[Slack], il suffit de la copier sur le RamDisk, et ensuite on pourra changer de CD-ROM sans problèmes.

3.2. Logiciels clés

3.2.1. Links 2

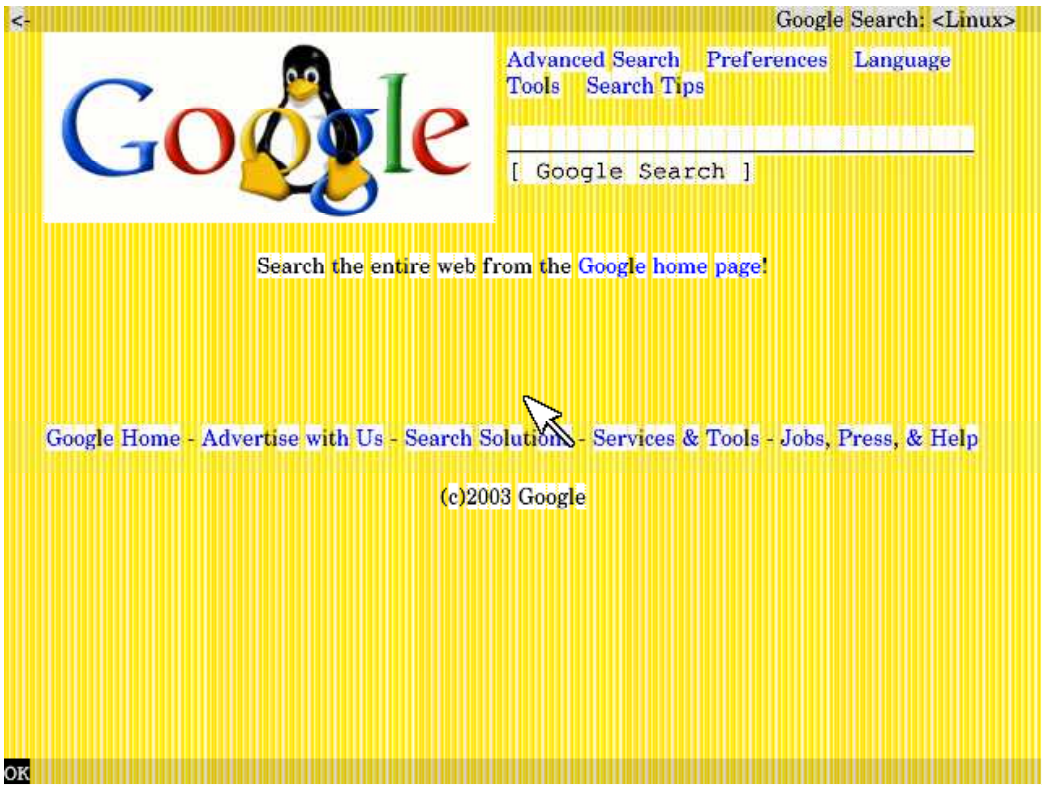
Links est un navigateur en mode texte tout comme Lynx, mais les développeurs de Twibright Labs l'ont étendu pour le mode graphique. Cette nouvelle version, nommée Links 2 [11], supporte plusieurs modes graphiques dont le FrameBuffer de Linux, le serveur X, la libSVGA, ainsi que DirectFB. Pour le moment c'est le pilote du FrameBuffer qui est utilisé sous Dream[Slack], en attend de tester l'implémentation du DirectFB.

Ils ont aussi rajouté le support du JavaScript, et c'est la raison principale au choix de ce navigateur en tant que coeur de l'interface de Dream[Slack]. En effet, le JavaScript permet de créer un clavier virtuel afin de remplir les champs d'un formulaire, ce qui est indispensable puisque le système est uniquement piloté avec Js2Mouse.

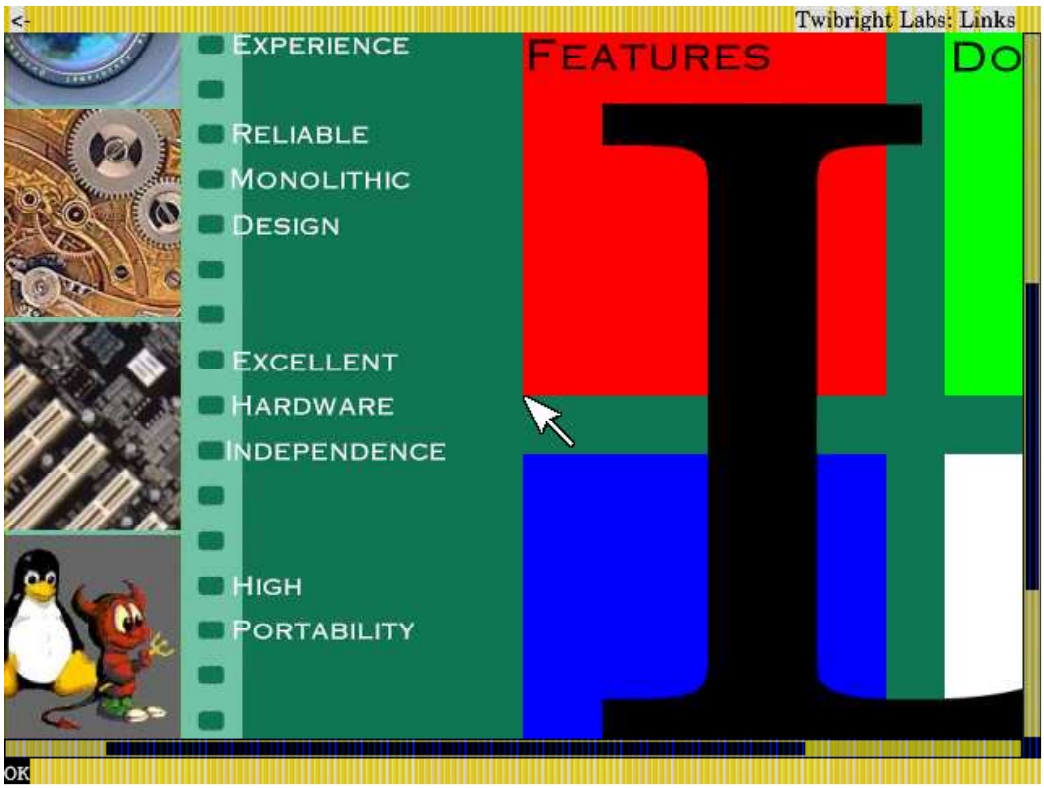
Tout aussi important, Links 2 est très peu gourmand en mémoire, et possède une vitesse de rendu impressionnante. De plus la police de caractère est issue d'images PNG anti-aliasées, ce qui offre un confort visuel très appréciable.

Lors des premiers testes, les couleurs du fond bavaient, mais cela ne se produisait pas sur les images. Un développeur de la mailing-list de LinuxDC émit l'hypothèse que le code du rendu des images et du rendu du fond étaient différents, et que l'accès au FrameBuffer devait se faire seulement à travers un (int *) ou un (short *) et non un (char *). Il suffit donc de le vérifié dans les sources de Links.

Exemple de Links 2 sur DreamCast, avec couleurs du fond baveuses :



Autre exemple de Links 2 sur DreamCast :



Par commodité, les captures d'écran suivantes ont été prises sur PC, car il me faut 11 minutes pour en prendre une sur DreamCast (chargement par le port série).

3.2.2. Boa

Boa [12] est un serveur HTTP mono-processus très léger, le système de connexion multiples est géré de manière interne. Cela ne l'empêche pas d'avoir de très bonnes performances, et même supérieures à celle d'Apache.

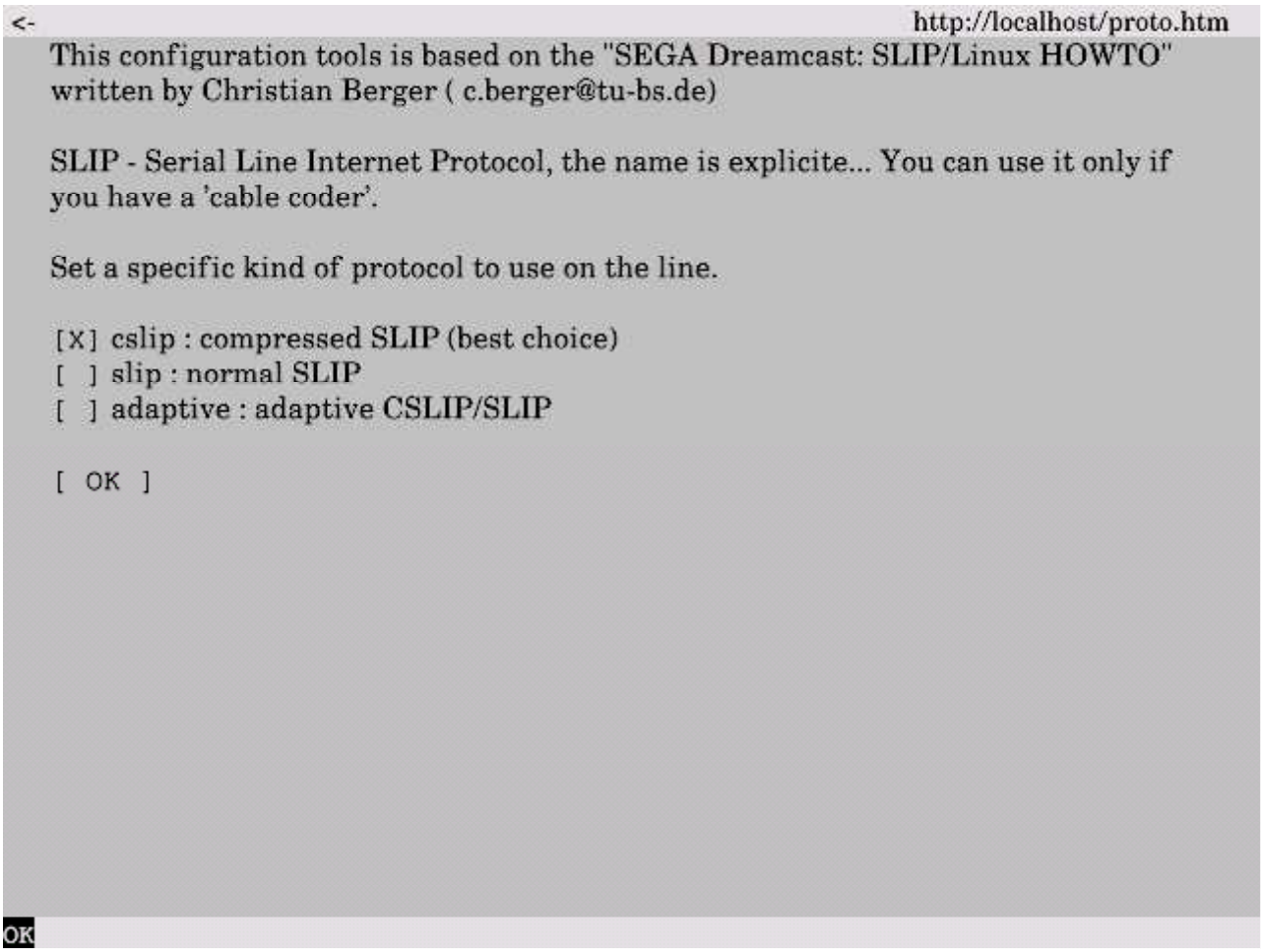
Le faite que Boa gère le script CGI fut primordiale, car cela permet d'exécuter des commandes sur la DreamCast.

3.3. Applications

3.3.1. Exemples

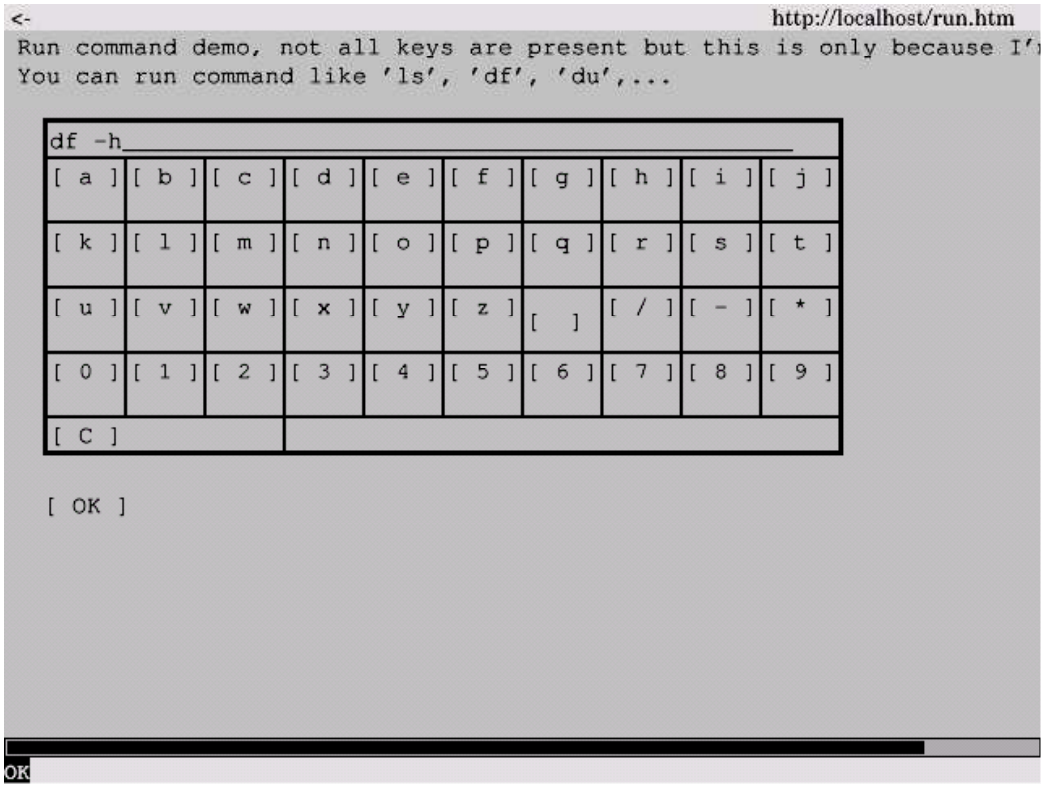
Voici un exemple d'application que l'on peut trouver sur Dream[Slack] : la configuration d'un réseau SLIP [13]. Ce dernier permet d'avoir une interface réseau grâce au port série de la DreamCast, ce qui signifie qu'il faille posséder le câble adéquat. Le but de la Dream[Slack] étant d'être utilisable par tous possesseurs 'traditionnels' de DreamCast, cet outil de configuration n'a qu'une valeur de démonstration, et cela permet d'être prêt lorsque le pilote du modem sortira puisque l'interface sera similaire.

Exemple de l'interface de configuration du SLIP pour Dream[Slack]:

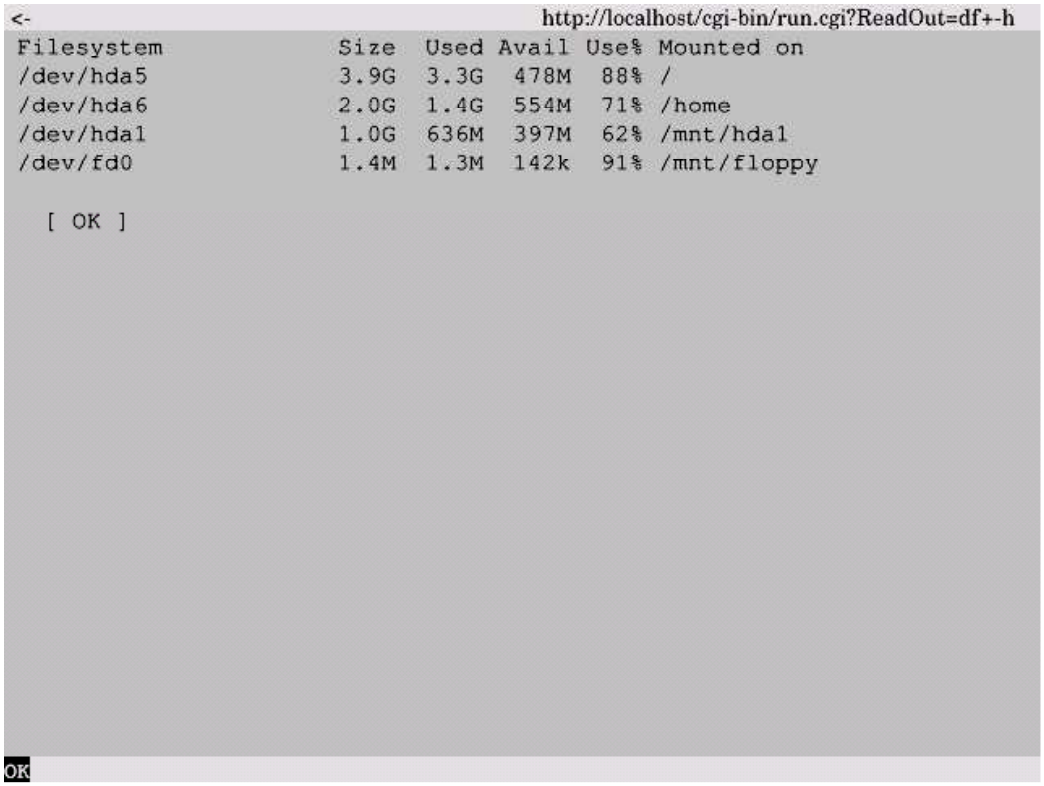


On peut même laisser à l'utilisateur le pouvoir de lancer ces propres commandes, cela peut être dangereux car s'il existe une connexion réseau, une personne mal intentionné pourrait exécuter des commandes arbitraires. Pour le moment Dream[Slack] ne s'occupe pas de la sécurité, mais dés qu'elle sera opérationnelle, ce point sera traité.

Exemple de clavier virtuel en JavaScript pour Links 2 :



Résultat de la commande exécuter sur l'hôte local :



3.3.2. Implémentation de l'interface

Voici des exemples de l'implémentation de l'interface. Ici je présente la page où l'utilisateur doit saisir l'adresse IP de sa DreamCast lors de la configuration du SLIP.

Fichier *ip-dc.htm* :

```
<html>
<head>
 <meta http-equiv="content-type"
content="text/html; charset=ISO-8859-1">
 <title></title>
</head>
<body>

Set your DreamCast serial interface IP, for example 192.168.0.2<br><br>

<FORM NAME="Keypad" ACTION="/cgi-bin/ifconfig.cgi">
                        <input type=hidden name=argument value="ip-dc">
<TABLE BORDER="0" WIDTH="50" HEIGHT="60" CELLPADDING="1" CELLSPACING="2">
<CAPTION ALIGN="top">
</CAPTION>

<TR>
        <TD COLSPAN="3" ALIGN="MIDDLE"><INPUT NAME="ReadOut" TYPE="Text" SIZE="24"
WIDTH="100%"></TD>
</TR>
<TR>
        <TD><INPUT NAME="btnSeven" TYPE="Button" VALUE="  7  " onClick="NumPressed(7)"></TD>
        <TD><INPUT NAME="btnEight" TYPE="Button" VALUE="  8  " onClick="NumPressed(8)"></TD>
        <TD><INPUT NAME="btnNine" TYPE="Button" VALUE="  9  " onClick="NumPressed(9)"></TD>
</TR>

<TR>
        <TD><INPUT NAME="btnFour" TYPE="Button" VALUE="  4  " OnClick="NumPressed(4)"></TD>
        <TD><INPUT NAME="btnFive" TYPE="Button" VALUE="  5  " OnClick="NumPressed(5)"></TD>
        <TD><INPUT NAME="btnSix" TYPE="Button" VALUE="  6  " OnClick="NumPressed(6)"></TD>
</TR>
<TR>
        <TD><INPUT NAME="btnOne" TYPE="Button" VALUE="  1  " OnClick="NumPressed(1)"></TD>
        <TD><INPUT NAME="btnTwo" TYPE="Button" VALUE="  2  " OnClick="NumPressed(2)"></TD>
        <TD><INPUT NAME="btnThree" TYPE="Button" VALUE="  3  " OnClick="NumPressed(3)"></TD>
</TR>
<TR>
        <TD><INPUT NAME="btnZero" TYPE="Button" VALUE="  0  " OnClick="NumPressed(0)"></TD>
        <TD><INPUT NAME="btnComma" TYPE="Button" VALUE="  .  " OnClick="NumPressed('.')"></TD>
        <TD><INPUT NAME="btnClear" TYPE="Button" VALUE="  C  " onClick="Clear()"></TD>
</TR>
</TABLE>
<INPUT  TYPE="SUBMIT" VALUE="OK">
</FORM>

<SCRIPT SRC="/keypad.js">
</SCRIPT>

</body>
</html>
```

Comme on le voit, c'est le script *keypad.js* qui s'occupe du clavier virtuel :

Fichier *keypad.js* :

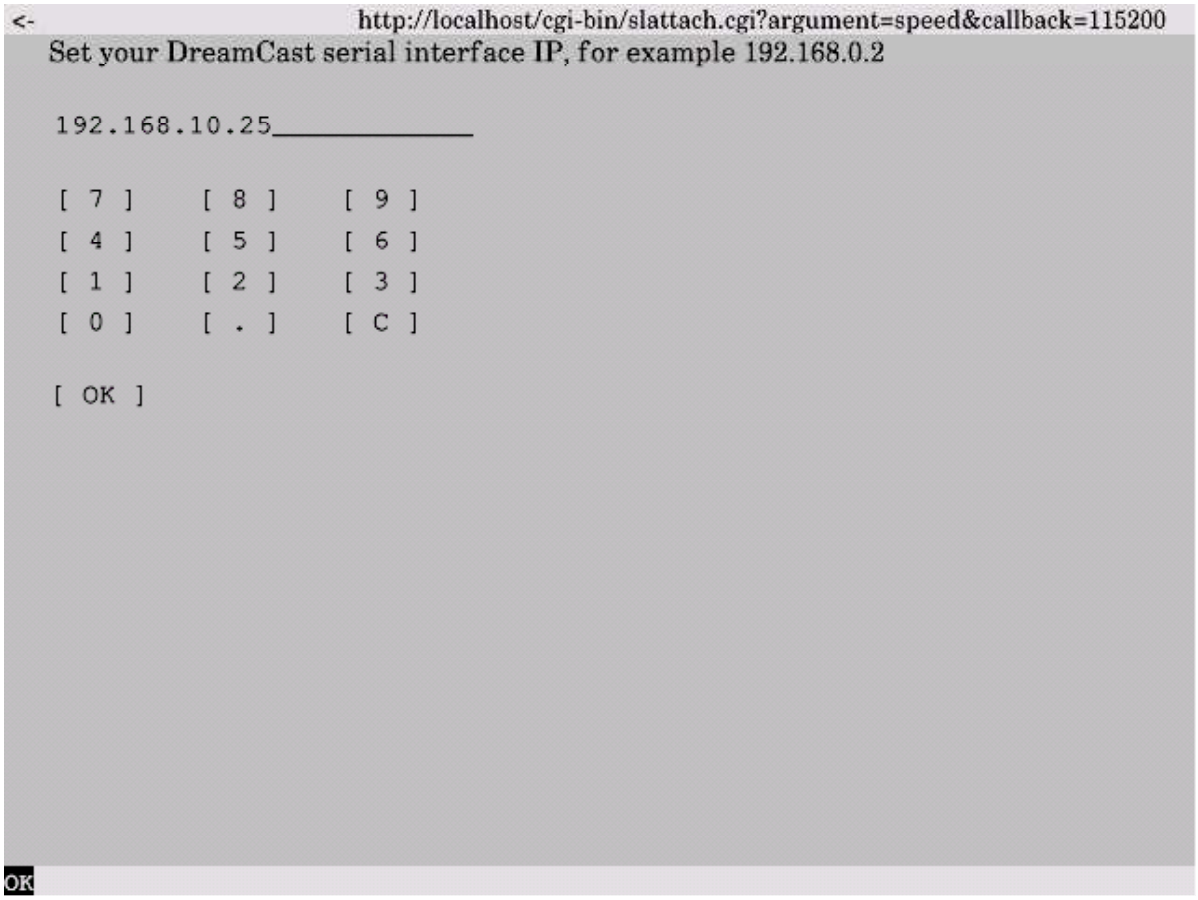
```
<!--
  // Module-level variables
  var FKeyPad = document.Keypad;
  var FlagNewNum = false;  // Flag to indicate a new number (operand) is being entered

  function NumPressed (Num)
  {
              if (FlagNewNum)
    {
                        FKeyPad.ReadOut.value  = Num;
      FlagNewNum = false;
    }
    else
    {
        FKeyPad.ReadOut.value += Num;
    }
  }

  function Clear ()
  {
    // Remove current number and reset state
    FKeyPad.ReadOut.value = "";
    FlagNewNum = true;
  }

//-->
```

Résultat :



Je ne détaillerais pas les scripts CGI qui s'occupent de la configuration SLIP, car ils sont un peu complexes. En revanche, le script *run.cgi* est beaucoup plus simple, nous avons déjà vu le résultat d'une exécution précédemment lors des exemples d'applications

Fichier *run.cgi* :

```
#!/bin/sh

echo "Content-type: text/html"
echo ""

echo "<html>"
echo "<head>"
echo "  <meta http-equiv=\"content-type\" content=\"text/html; charset=ISO-8859-1\">"
 echo "<title></title>"
echo "</head>"
echo "<body>"

TMP=/tmp/

OPTS=`echo $QUERY_STRING | sed 's/&/ /g'`
COMMAND=`echo $OPTS | sed 's/=/ /g' | awk '{print $2}' |  sed 's,%,\\\x,g' | sed 's/+/ /g'`

#exécute la commande et affiche sa sortie standart sur une page HTML
echo "<pre>"
$COMMAND
echo "</pre>"

if [ $? != 0 ]
then
        echo "ERROR : "
        cat $TMP/tmpERRds
        rm $TMP/tmp*ds
        cat ../failed.htm
        exit
fi

echo "<br>"
echo "<FORM NAME=\"Keypad\" ACTION=\"/index.html\">"
echo "     <INPUT  TYPE="SUBMIT" VALUE=\"OK\">"
echo "</FORM>"

echo "</body>"
echo "</html>"
```

3.3.3. Idées et évolution

Le problème lié à GPM peut être contourné de différentes manières, je pourrais tracer l'exécution de ce logiciel et de ceux qui l'utilisent pour trouver d'où vient le problème. Dans le cas contraire, je pourrais intégrer directement un support du joystick à Links 2.

Schéma actuel :

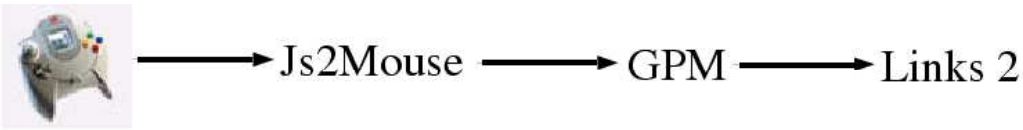
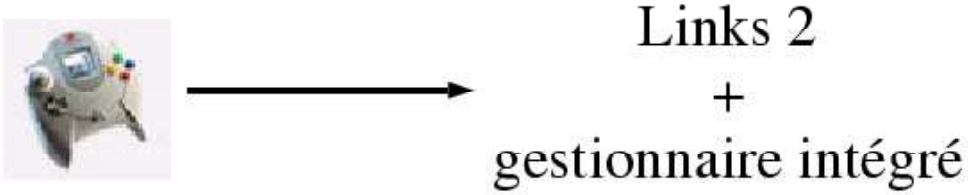


Schéma de secours :



Le support du VMU et de son système de fichiers venant d'être intégrés au noyau Linux, Kurdy présenta l'idée de pouvoir copier des sauvegardes de jeux à partir du CD-ROM de Dream[Slack] vers le VMU.

De plus Kurdy a commencé à créer une jolie interface HTML intuitive et complète.

Pour le moment, les formulaires sont remplis grâce à un clavier virtuel écrit en JavaScript, mais lorsque Linux pour DreamCast sera capable d'utiliser le modem, il faudra trouver une autre solution pour pouvoir naviguer sur Internet. D'où l'idée d'implémenter directement un clavier virtuel dans Links 2, comme le font les navigateurs officiels de SEGA sur DreamCast.

4. Annexes

4.1. Références

[1] http://www.kernel.org
[2] http://www.sega.com
[3] http://linuxsh.sourceforge.net
[4] http://linuxdc.sourceforge.net
[5] http://www.xfree86.org
[6] http://cadcdev.sourceforge.net
[7] http://freshmeat.net/projects/gpm
[8] http://www.billgatliff.com
[9] http://www.m17n.org/linux-sh/dreamcast
[10] http://www.sh-linux.org
[11] http://atrey.karlin.mff.cuni.cz/~clock/twibright/links
[12] http://www.boa.org
[13] http://www-public.tu-bs.de:8080/%7Ey0018536/dc/sl0.html

4.2. Bibliographie

- *Programmation Unix*, Warren W. Gay, Campus Press.

- *Pilotes de périphériques sous Linux 2e édition*, Alessandro Rubini et Jonathan Corbet, O'Reilly.

- *Linux embarqué*, Pierre Ficheux, Eyrolles.

- *Le noyau Linux*, Daniel P. Bovet et Marco Cesati, O'Reilly.

4.3. Étude de DCLinux (en Anglais)

Voici un texte que j'ai rédigé en Anglais (ce rapport devait l'être aussi) dans le cadre de ce T.E.R. Il s'agit de la dissection de la séquence de démarrage de DCLinux-010605.

4.3.1. ECos as boot loader.

-Load into memory /boot/vmlinux @ 0x8c210000
-Load into memory /boot/initrd.gz @ 0x8c400000
-Exec Linux kernel with "mem=16M init=/busybox /etc/profile" as arguments

4.3.2. System on the ramdisk.

Be careful, all the file names relate to the ramdisk, NOT to the gdrom !!!

-'busybox' is a "ELF 32-bit LSB executable, Hitachi SH, version 1, statically linked, stripped", so it don't need any libs...

-'/etc/profile' is a symlink to '/gdrom/etc/profile'.
There is another file named 'profile' into '/gdrom', but I don't know why...
The '/gdrom/etc/profile' file contains the 'mount -a' command, that causes all file systems mentioned in '/etc/fstab' to be mounted as indicated :
-'etc/fstab' is a symlink to '/gdrom/etc/fstab' that contains :
proc /proc proc defaults 0 0
/dev/gdrom /gdrom iso9660 ro 0 0
So the proc_fs is working and the '/gdrom' entry from ramdisk is NOW the root of the gdrom... Be careful...
The '/gdrom/etc/profile' file contains, finally, the command 'exec /sbin/init', so the control is given to the system on the gdrom.

4.3.3. System on the gdrom

Be careful, some file names (ie : '/gdrom') are overwrite by the root of the gdrom !!!

-Read '/etc/inittab' that is NOT a symlink to '/gdrom/etc/initab', but this last file exist, why ??? Here comes the more importants lines :

```
# The default runlevel.
    id:2:initdefault:
[...]
# This is run first except when booting in emergency (-b) mode.
    si::sysinit:/etc/init.d/rcS
[...]
# /etc/init.d executes the S and K scripts upon change
    l2:2:wait:/etc/init.d/rc 2
[...]
# /sbin/getty invocations for the runlevels.
# The "id" field MUST be the same as the last characters of the device (after "tty").
# Format:  <id>:<runlevels>:<action>:<process>
    1:2345:respawn:/sbin/getty 38400 tty1
[...]
# Example how to put a getty on a serial line (for a terminal)
    SC1:23:respawn:/sbin/getty -L ttsc/1 115200 vt100
```

-Fisrt, this is the '/etc/init.d/rcS' script that is invoked, this main role is to launch script '/etc/rcS.d/S??' :*

S01devfsd -> ../init.d/devfsd
S05keymap.sh -> ../init.d/keymap.sh
S10checkroot.sh -> ../init.d/checkroot.sh
S20modutils -> ../init.d/modutils
S30checkfs.sh -> ../init.d/checkfs.sh
S30procps.sh -> ../init.d/procps.sh
S35devpts.sh -> ../init.d/devpts.sh
S35mountall.sh -> ../init.d/mountall.sh
S39ifupdown -> ../init.d/ifupdown
S40hostname.sh -> ../init.d/hostname.sh
S40networking -> ../init.d/networking
S41portmap -> ../init.d/portmap
S45mountnfs.sh -> ../init.d/mountnfs.sh
S48console-screen.sh -> ../init.d/console-screen.sh
S50hwclock.sh -> ../init.d/hwclock.sh
S55bootmisc.sh -> ../init.d/bootmisc.sh
S55urandom -> ../init.d/urandom
S70nviboot -> ../init.d/nviboot
S90nvi-m17n -> ../init.d/nvi-m17n

-Then '/etc/init.d/rc 2' run scripts found in '/etc/rc2.d' :
S10ipchains -> ../init.d/ipchains
S10sysklogd -> ../init.d/sysklogd
S11klogd -> ../init.d/klogd
S14ppp -> ../init.d/ppp
S20gpm -> ../init.d/gpm
S20inetd -> ../init.d/inetd
S20logoutd -> ../init.d/logoutd
S20makedev -> ../init.d/makedev
S99rmnologin -> ../init.d/rmnologin
s19nfs-common -> ../init.d/nfs-common
s20nfs-kernel-server -> ../init.d/nfs-kernel-server
s89atd -> ../init.d/atd
s89cron -> ../init.d/cron
s91apache -> ../init.d/apache

-And finally, '/sbin/init' runs getty...

4.3.4. Result

-Look at :

```
 09:02:59 up 2 min,  1 user,  load average: 0.28, 0.23, 0.09
17 processes: 16 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  0.6% user,  0.4% system,  0.0% nice,  99.0% idle
Mem:    14616K total,    11516K used,    3100K free,    2316K buffers
Swap:       0K total,       0K used,       0K free,    5500K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT %CPU %MEM   TIME COMMAND
  183 root      18   0  1084 1084   876 R     0.9  7.4   0:00 top
    1 root       9   0   596  596   512 S     0.0  4.0   0:01 init
    2 root       9   0     0    0     0 SW    0.0  0.0   0:00 keventd
    3 root       9   0     0    0     0 SW    0.0  0.0   0:00 kswapd
    4 root       9   0     0    0     0 SW    0.0  0.0   0:00 kreclaimd
    5 root       9   0     0    0     0 SW    0.0  0.0   0:00 bdflush
    6 root       9   0     0    0     0 SW    0.0  0.0   0:00 kupdated
    7 root       9   0     0    0     0 SW    0.0  0.0   0:00 kmapled      <-for DreamCast hotplug devices (looks like USB)
   39 root       9   0   900  900   668 S     0.0  6.1   0:00 devfsd
   91 root       8   0   652  652   556 S     0.0  4.4   0:00 pump     <-configure network interface BOOTP or DHCP protocol
   96 daemon     9   0   524  524   420 S     0.0  3.5   0:00 portmap
  149 root       9   0   944  944   796 S     0.0  6.4   0:00 syslogd
  152 root       9   0   604  604   456 S     0.0  4.1   0:00 klogd
  158 root       9   0   572  572   488 S     0.0  3.9   0:00 gpm
  165 root       9   0   816  816   704 S     0.0  5.5   0:00 inetd
  177 root       9   0   540  540   456 S     0.0  3.6   0:00 getty
  178 root       9   0  1384 1380  1116 S     0.0  9.4   0:01 bash
```

4.4. Construction de Dream[Slack] (en Anglais)

Voici un autre document en Anglais, toujours rédigé dans le cadre de ce TER, traitant de la construction de Dream[Slack], la configuration des logiciels et la création d'un CD ne sont pas présents.

Notes:
^^^^^

Respecte ordre

There are several versions of packages : NOT all permutations (dependences) are tested, only newer versions are know to work. Other informations are here only if you want to install other versions, I explain some tricks that can help you...

Set environment variables (except when is specified to don't):

```
export TARGET=sh4-linux
export PREFIX=/usr/local/DreamCast/sh4-linux
export PATH=${PREFIX}/bin:${PATH}
export LD_LIBRARY_PATH=${PREFIX}/lib
export BINDIR=$PREFIX/BIN
```

If you've just install tools chains, you must create a directory :

```
mkdir $PREFIX/BIN
```

Installation :
*^^^^^^^^^^*
AS ROOT !!!

install tools chains :
See Bill Gatliff's article 'emblindc' at http://www.billgatliff.com, if you are too lazy (you're wrong, this article is wonderful) :
cd /
wget http://www710.univ-lyon1.fr/~c-vinc02/downloads/Linux-DreamCast-Xdev-i386.tar.gz
tar -xvzf Linux-DreamCast-Xdev-i386.tar.gz

install ncurses :
(ncurses-5.3)
First installation will NOT include 'gpm' support !!!
./configure --with-shared --with-libtool --host=i386 --target=$TARGET --prefix=$PREFIX --bindir=$BINDIR --with-build-cc=/usr/bin/gcc --with-build-libs=/lib/ld-linux.so.2
make
make install
That's all right...

(ncurses version <5.3)
If you've got an error with 'make_keys' and 'make_keys', here comes a trick :
From scratch, try to compile ncurses natively, so DON'T set environment variables :
./configure --with-gpm --with-shared --with-libtool
*make | grep make_*
When you see the succes building of make_keys and make_hash (few seconds), you can stop compiling.
Then unpack another package for cross compilation, set environment variables.
Remove lines after the targets 'make_keys' and 'make_hash' in 'ncurses/Makefile.in', it should be seems like this :
make_keys$x :

make_hash$x :

Then copy the 'ncurses/make_keys' and 'ncurses/make_hash' binaries from your host build to the cross compilation build.
Let's go :
./configure --with-shared --with-libtool --host=i386 --target=$TARGET --prefix=$PREFIX --bindir=$BINDIR
make
make install
That's all right...

install gpm :
(gpm-1.20.0, gpm-1.19.3)
This installation will include 'ncurses' support !!!
./configure --prefix=$PREFIX --bindir=$BINDIR --with-curses
make
make install (you may got an error with man page compressed or some thing like that, ignore it)

(gpm-1.19.6)
If you've got an error when 'make' (gpm-1.19.6)
make (maybe you've get an error with gpm.dvi, ignore it)
cd src
mkdir $PREFIX/sbin
make install (you may still got an error, ignore it)
That's all...

reinstall ncurses :
(ncurses-5.3)

Second installation WILL include 'gpm' support !!!

```
make clean
./configure --with-gpm --with-shared --with-libtool --host=i386 --target=$TARGET
--prefix=$PREFIX --bindir=$BINDIR --with-build-cc=/usr/bin/gcc --with-build-libs=/lib/ld-linux.so.2
make
make install
```

That's all right...

(ncurses version <5.3)

If you've got an error with 'make_keys' and 'make_keys', here comes a trick :
From scratch, try to compile ncurses natively, so DON'T set environment variables :

```
./configure --with-gpm --with-shared --with-libtool
make | grep make_
```

When you see the succes building of make_keys and make_hash (few seconds), you can stop compiling.
Then unpack another package for cross compilation, set environment variables.
Remove lines after the targets 'make_keys' and 'make_hash' in 'ncurses/Makefile.in', it should be seems like this :

```
make_keys$x :

make_hash$x :
```

Then copy the 'ncurses/make_keys' and 'ncurses/make_hash' binary from your host build to the cross compilation build.
Let's go :

```
./configure --with-gpm --with-shared --with-libtool --host=i386 --target=$TARGET
--prefix=$PREFIX --bindir=$BINDIR
make
make install
```

That's all right...

install zlib :
(zlib-1.1.4)

First, we install static library :

```
./configure --prefix=$PREFIX
make
make install
```

Then dynamic library :

```
./configure --prefix=$PREFIX --shared
make
make install
```

That's all right...

install libpng :
(libpng-1.2.5)

There is NO script 'configure', instead you must copy/edit makefile :

```
cp scripts/makefile.linux ./
```

Replace 'prefix=/usr/local/' with 'prefix=/usr/local/DreamCast/sh4-linux/' in 'makefile.linux'
Replace 'BINPATH=$(prefix)/bin' with 'BINPATH=$(prefix)/BIN' in 'makefile.linux'

```
make -f makefile.linux
make -f makefile.linux install
```

That's all right...

install libjpeg :
(jpeg-6b)

This librairy need libtool in its directory, why ?!?

cp `which libtool` .
./configure --prefix=$PREFIX --bindir=$BINDIR --enable-shared
make
make install

That's all right...

install libtiff :
(tiff-v3.6.0-beta)

There is the same problem: like 'ncurses < 5.3', host and target are considered the same !!!
From scratch, try to compile 'libtiff' natively, so DON'T set environment variables :

./configure
make

Then unpack another package for cross compilation, set environment variables.
Remove this line after the target '${SRCDIR}/tiffvers.h' in 'libtiff/Makefile.in' :

${CC} -o mkversion ${CFLAGS} ${SRCDIR}/mkversion.c

Remove this line after the target 'tif_fax3sm.c' in 'libtiff/Makefile.in' :

${CC} -o mkg3states ${CFLAGS} ${SRCDIR}/mkg3states.c

Then copy the 'libtiff/mkversion' and 'libtiff/mkg3states' binaries from your host build to the cross compilation build.
Let's go :

./configure --with-shared --prefix=$PREFIX

Change default bindir and mandir :

Directory to install tools [/usr/local/DreamCast/sh4-linux/bin]? /usr/local/DreamCast/sh4-linux/BIN
Directory to install manual pages [/usr/local/man]? /usr/local/DreamCast/sh4-linux/man

Ok :

make
make install

(tiff-v3.5.7)

I wasn't able to compile tiff-3.5.7 as a dynamic librairy, unless to compile it as static :
From scratch, try to compile 'libtiff' natively, so DON'T set environment variables :

./configure
make

Then unpack another package for cross compilation, set environment variables.
Remove this line after the target '${SRCDIR}/tiffvers.h' in 'libtiff/Makefile.in' :

${CC} -o mkversion ${CFLAGS} ${SRCDIR}/mkversion.c

Remove this line after the target 'tif_fax3sm.c' in 'libtiff/Makefile.in' :

${CC} -o mkg3states ${CFLAGS} ${SRCDIR}/mkg3states.c

Then copy the 'libtiff/mkversion' and 'libtiff/mkg3states' binaries from your host build to the cross compilation build.
Let's go :

./configure --with-shared --target=$TARGET --prefix=$PREFIX

Change default bindir and mandir :

Directory to install tools [/usr/local/DreamCast/sh4-linux/bin]? /usr/local/DreamCast/sh4-linux/BIN
Directory to install manual pages [/usr/local/man]? /usr/local/DreamCast/sh4-linux/man

Ok :

make
make install

That's all right...

install bash :

[not write yet, but it works !]

install links2.1 :
(links-2.1-pre7 or newer)

This is a excellent program, so there is no problem !!!

```
./configure --prefix=$PREFIX --bindir=$BINDIR --enable-javascript --enable-graphics --without-x
make
make install
```

install boa :
(boa-0.94.13)

Edit 'configure' and remove 'exit 1;' at the end of the line that contain 'cross compiling'.

```
./configure --prefix=$PREFIX --bindir=$BINDIR
make
```