Introduction à uClinux

J.-M Friedt, S. Guinot
Association Projet Aurore

4 mars 2005

UNIVERSITÉ DE FRANCHE-COMTÉ
PROJET AURORE
DIFFUSION DE LA CULTURE ET DE L'INFORMATION SCIENTIFIQUE ET TECHNIQUE

Principales étapes à réaliser

- Mise en œuvre du matériel (carte uCdimm 5272)
- Installation des outils de compilation
- Mise en place d'un environnement de développement
- Au travail ...

Ces transparents et programmes :
`http://projetaurore.assos.univ-fcomte.fr/uclinux`
Notice complète : `http://friedtj.free.fr/uclinux.pdf`

Quelques définitions

- **Noyau** : scheduler, gestion de la mémoire, drivers pour les accès matériels, systèmes de fichiers
- **Linux** : noyau (kernel) opensource, disponible à `http://www.kernel.org` [1, 2]
- **Distribution** : programmes et méthodes de distribution annexes (=unix)

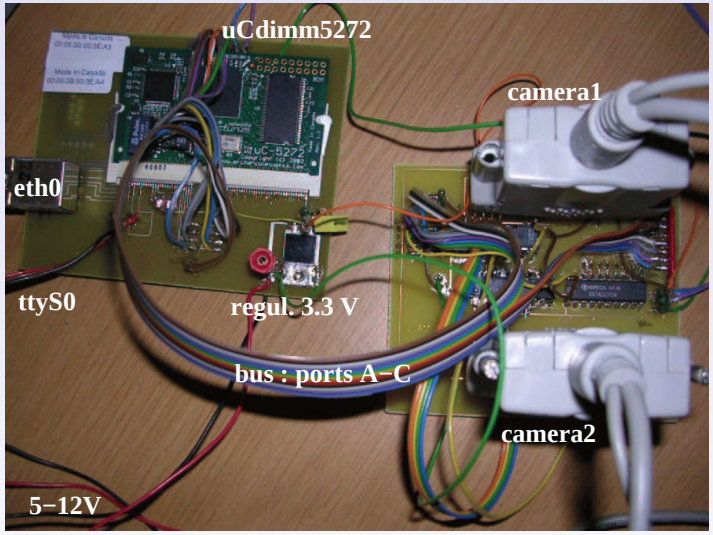
Disponibilité des sources $\Rightarrow$ adaptable à nos besoins, recompilable pour diverses architectures

[1] L. Torvalds, D. Diamond *Just for Fun : The Story of an Accidental Revolutionary*, HarperCollins (1999)

[2] M. Welsh, M.K. Dalheimer, T. Dawson et L. Kaufman *Le système Linux*, 4ème édition, O'Reilly Ed. (2003)

La carte uCdimm 5272

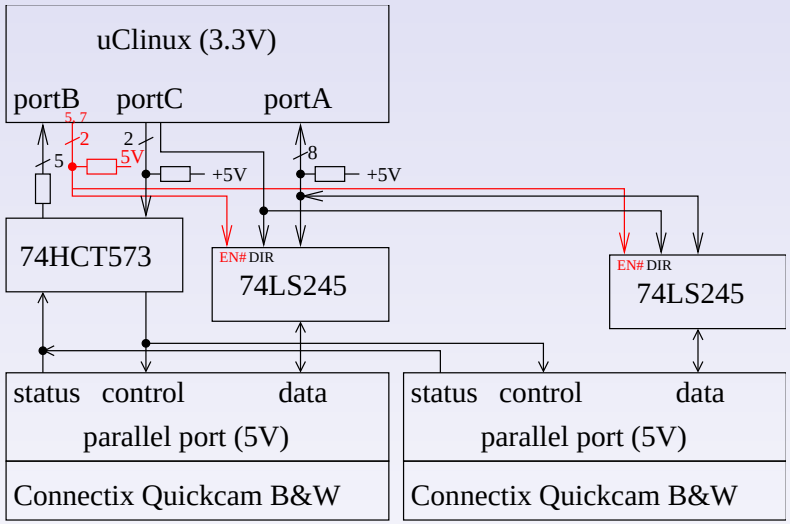
- Processeur Coldfire 5272 à 66 MHz, 8 MB RAM, 4 MB flash
- Dimensions réduites, puissance requise minimale (1 W)
- Régulateur de tension : 3.3 V
- Nombreux périphériques :
  - 2 ethernet,
  - 2 RS232 ($\pm$5 V),
  - 20 bits bidirectionnels,
  - 2 PWM,
  - SPI
- Accès au bus de données et d'adresses (décodeur d'adresse : chip select)
- Préinstallé avec un noyau uclinux
- support SODIMM 144 broches



La carte uCdimm 5272

Électronique d'interfaçage CMOS 3.3 V/TTL :

- résistance de pull-up en sortie
- résistance de limitation de courant en entrée
- émulation du port parallèle de PC (8 bits bidirectionnels, 5 bits en entrée, 3 bits en sortie, 3 bits de contrôle direction/arbitrage)



Bootloader :

- communique avec le PC par RS232 (9600, N81) : `minicom`
- exécuter le système d'exploitation en flash, charger une image du système (RS232)

Les outils de cross-compilation

- Archive précompilée pour processeur Intel
- Compilation de `gcc+binutils+`divers outils (`genromfs`, `STLport`) à destination de `m68k-elf` [3]
- accès au résultat de la compilation : on monte le système de fichiers par NFS
  - sur le PC, `/etc/exports` inclut `/home 172.16.1.19(rw)`
  - le PC fait tourner les serveurs NFS (`rpc.nfsd`, `rpc.mountd`)
  - sur uClinux :
    définition de son adresse IP : `ifconfig eth0 172.16.1.19`
    définition éventuelle de la table de routage :
    `route add default gw 172.16.1.1`
  - une connexion `telnet` remplace la liaison RS232
  - monter le système de fichiers du PC :
    `mount -o nolock,mountvers=2 172.16.1.1 :/home/jmfriedt /mnt`

[3] `http://www.uclinux.org/pub/uClinux/uclinux-elf-tools/gcc-3/build-uclinux-tools.sh`

Spécificités de uClinux (www.uclinux.org)

- supporte une vaste gamme d'architectures (H8, SH, MIPS, x86, i960, ARM, m68k)
- fonctionne en l'absence de gestionnaire de mémoire (MMU)
- nécessite une quantité minime de mémoire (noyau de taille réduite)
- fournit un scheduler pour le multitache
- fournit une couche de communication réseau (IP, TCP, UDP)
- supporte un grand nombre de systèmes de fichiers
- émulation logicielle du calcul flottant en l'absence de FPU

Compilation de uClinux

Obtention d'une archive du noyau `uclinux` sur `http://www.uclinux.org/pub/uClinux/dist/`

- inclut les noyaux 2.4, 2.6,
- les configurations par défaut pour un grand nombre de cartes (voir `vendors/Arcturus/uC5272`)
- libc et diverses librairies (compressions, cryptographie, curses ...)
- les sources de nombreuses applications (ls, mv ... busybox, boa, ...)

Configuration du noyau pour notre carte : `make menuconfig`

- sélection de la carte (configuration constructeur : Arcturus, uC5272)
- choix de la librairie `uClibc` et `Customize Kernel Settings`
- adaptation à notre matériel (`Processor Type and Features` : 8 MB flash, 66 MHz ; `Block devices` : 4 MB flash)
- choix des modules du noyau
- choix des applications et des librairies si `Customize Vendor/User Settings` est sélectionné

Compilation par `make dep && make`, image iso de la flash par `make image`

L'environnement de travail

On compile à destination de m68k-elf (puis flat memory model) sur processeur Intel (ELF)
Exemple de Makefile pour compiler nos applications

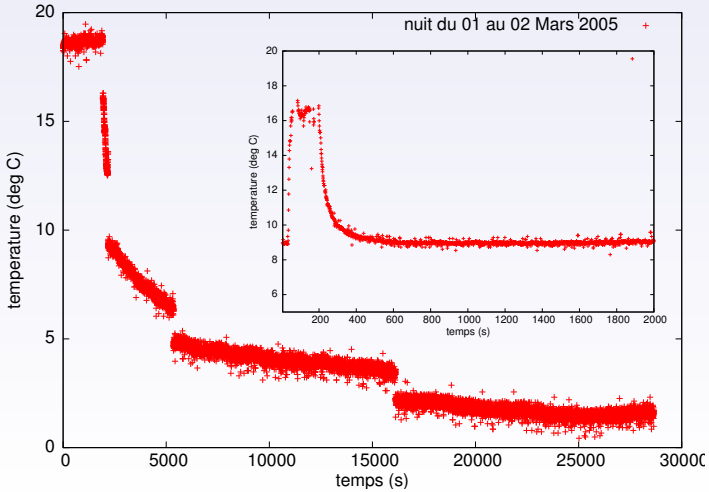
```
PATH := $(PATH):/usr/bin:/usr/local/bin
CC = m68k-elf-gcc
EXEC = pwm
OBJS = pwm.o
UCPATH = /home/jmfriedt/uclinux/uClinux-dist
CINCL = -I$(UCPATH)/lib/libc/include -fno-builtin -msep-data -I$(UCPATH)/linux-2.4.x/include
LDLIB = -L$(UCPATH)/lib/libc -L$(UCPATH)/lib/libm
CFLAGS = -m5307 -DCONFIG_COLDFIRE -Os -g -fomit-frame-pointer -Dunix -D__uClinux__ -DEMBED $(CINCL)
LDFLAGS = $(CFLAGS) -Wl,-elf2flt -Wl,-move-rodata $(LDLIB)

all: $(EXEC)

$(EXEC): $(OBJS)
$(CC) $(LDFLAGS) -o $@ $(OBJS) $(LDLIBS) -lc -lm

clean:
rm -f $(EXEC) *.elf *.gdb *.o
```

Premier exemple : `Hello World` (stdout sur le port série, calcul flottant)

Capture de télémétrie

Communication avec un microcontroleur par le port RS232 (9600, N81)

- le port série est monopolisé par le terminal qui lui est lié (voir `/etc/inittab` qui appelle `agetty`)
- il nous faut tuer ce processus et lancer notre programme : `kill xx && ./rs_rec`
- le programme de lecture du port série est *strictement* identique à celui utilisé sur PC (Intel)
  $\Rightarrow$ ouverture de `/dev/ttyS0`, définition de ses attributs par `tcsetattr()`, puis lecture/écriture (`read()` et `write`) sur ce desripteur de fichier (`open()`).
- problème d'avoir du RS232 en ± 5 V : il faut ajouter un MAX232



Exemple de mesure de température acquise au moyen d'un Analog Devices ADuC814 et transmis à 9600 bauds à l'uclinux par RS232.

Accès aux ports généraux (GPIO)

Absence de MMU : on peut accéder à toutes les adresses, y compris celles des registres matériels du Coldfire (PEEK/POKE). Inutile de réserver l'accès au port (`ioperm()` sous linux).

Tous les registres d'accès au matériel sont indexés par rapport à MBAR (défini dans `asm/coldfire.h`) [4, p.6-4].

- Accès aux ports d'entrées-sorties [4, ch.17]
  en sortie
  ```
  *((volatile unsigned short *)(MCF_MBAR+MCFSIM_PCDDR))=0xe0ff ;
  ```
  en entrée
  ```
  valeur=*((volatile unsigned char*)(MCF_MBAR+MCFSIM_PCDAT)) ;
  ```
- Problème d'endianness entre Intel et Motorola : ne pas oublier les `hton` et `ntoh`

[4] MCF5272 Coldfire Integrated Microprocessor User's Manual (MCF272UM/D, rev 2, 03/2002,
`http://www.freescale.com/files/dsp/doc/ref_manual/MCF5272UM.pdf`

Accès aux PWM

Par exemple pour le contrôle de servo moteurs de modélisme :

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <asm/coldfire.h>           // defines MCF_MBAR
#include <asm-m68knommu/m5272sim.h> // defines PADDR
#include "webcam.h"

#define MCFSIM_PWCR0 (0xc0)
#define MCFSIM_PWWD0 (0xd0)

int main(int argc,char **argv)
{short i=0;char delai=25;

 if (argc>1) {delai=atoi(argv[1]);
 if ((delai>37)||(delai<13)) {printf("12<delai<37\n");delai=25;}
 else printf("delai=%d\n",delai);}
 if (argc>2) {i=atoi(argv[2]);printf("i=%d\n",i);}

 *((volatile unsigned char*)(MCF_MBAR+0+MCFSIM_PWCR0+i*4))=0xac;
 *((volatile unsigned char*)(MCF_MBAR+0+MCFSIM_PWWD0+i*4))=delai;
}
```

Application pratique

Objectif : transmettre des images stéréoscopiques depuis un ballon captif
Moyens : puissance de calcul (protocole TCP/IP, threads, compression jpeg)
Capteurs :

- caméras Connectix Quickcam noir et blanc (port parallèle, 1994)
- multiplexage des ports de l'uCdimm pour émuler deux ports parallèles
- ajouts possibles sur RS232 (ADC, GPS ...)

Architecture de l'application

Temps réel : un sytème est dit temps réel lorsque l'information après acquisition et traitement reste encore pertinente

$\longrightarrow$ le temps réel et uClinux :

- noyau préemptif
- ordonnanceur gérant la priorité des tâches

$\longrightarrow$ le temps réel et notre application :

- TCP
- multithreading
- compression jpeg

Thread : processus léger ou flot d'instructions

- tous les threads d'un processus partagent le même espace d'adressage
- paralléliser les tâches d'un processus

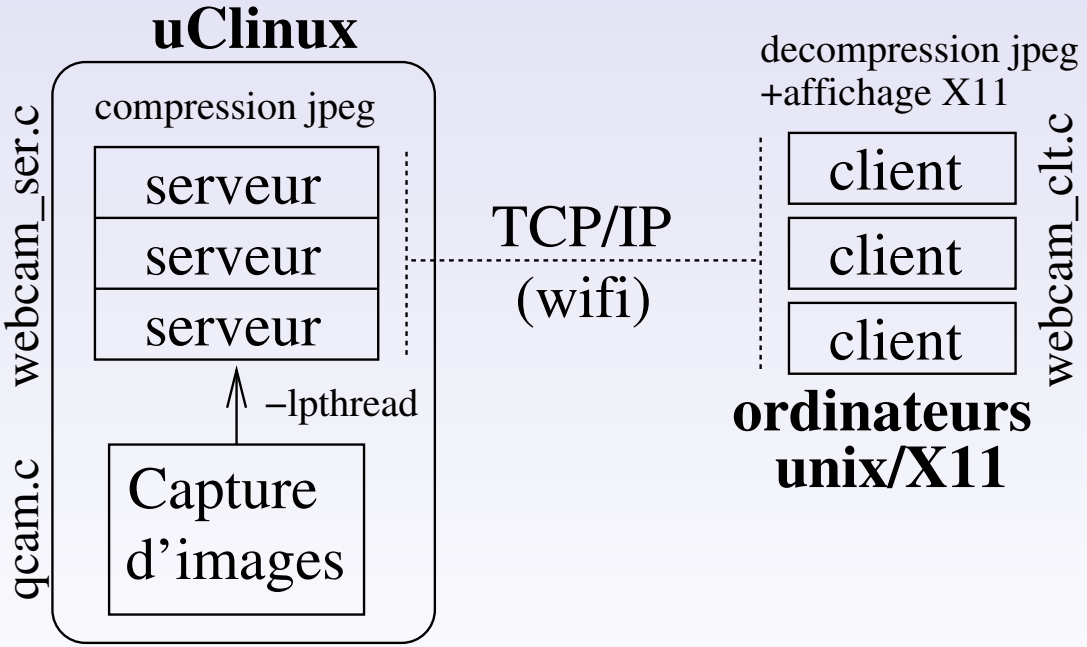
Algorithme

Extrait du code de webcam_ser.c

```
void *webcam_thread (void *args);
...
  if (first_accept)
  {
   if (pthread_create (&th_wc, NULL, webcam_thread, NULL) == -1)
      {
perror ("pthread_create ()");
        return (-1);
      }
      first_accept = 0;
  }
  pthread_mutex_lock (&mutex_clt_list);
  clt_list_sock[num] = clt_sock;
  pthread_mutex_unlock (&mutex_clt_list);
...
```

Architecture de l'application

- Serveur embarqué sur l'uCdimm (uclinux)
- Clients multiples (unix/X11) se connectent au serveur [5]
- Utilisation de TCP : mode connecté nous informe de la perte de liaison
- La bande passante la plus réduite limite le débit d'images



[5] http://www.linuxgazette.com/issue47/bueno.html

Résultats

324×243 pixels=78732 bytes/image
À 6 Mb/s (LAN), le transfert prend 0.1 s, mais ADSL à 128 kb/s ⇒ 5 s
Paramètre de compression : 50, temps d'exposition : 90

```
jmfriedt@home:~/test/uc_webcam$ ./webcam_clt grabber
connect to grabber
depth = 16
3:0:10564 6:1:6984
10:0:10227 10:1:6933
13:0:10219 16:1:6973
16:0:10217 19:1:6941
22:0:10182 22:1:6949
25:0:10137 28:1:6953
28:0:10179 31:1:6957
34:0:10128 34:1:6967
37:0:10172 40:1:6959
43:0:10197 43:1:6951
46:0:10189 50:1:6951
50:0:10192 53:1:7012
56:0:10133 56:1:6973
59:0:10167 62:1:6956
```

Conlusion

Nous avons démontré :

- la mise en œuvre matérielle d'une carte destinée à exécuter uclinux
- la mise en œuvre des outils de développement (noyau+applications)
- la capacité à interagir avec divers types de périphériques (RS232 pour microcontroleur, PWM pour servo moteur, GPIO pour caméra)
- la capacité à transférer des données à débit élevé
- la capacité à mettre en œuvre une puissance de calcul conséquente (compression jpeg) *v.s* puissance électrique consommée

Perspectives

Il nous reste à :

- gérer les interruptions matérielles et la communications SPI
- transmettre des données du/au processeur *via* les bus d'adresse et de données
- découvrir de nouvelles cartes (SSV DILNet5280 : plus puissante, plus pratique, moins chère)
- embarquer nos caméras sur un ballon captif/ballon sonde
- commande aéromodélisme (application au drone de l'ENSMM)

Sources d'inspiration :

`http://members.shaw.ca/sonde/` : lancer un planeur depuis un ballon (amateur, perdu)
`http://weather.ou.edu/~fgallag/glider/index.shtml` : lancer un planeur depuis un ballon (NOAA, fini ?)
`http://www.aerosonde.com/` : premier vol transatlantique (privé, NASA/envrionnement/militaire)

`http://www.linuxdevices.com/articles/AT4739871225.html` et `http://www.arctic.noaa.gov/gallery_np.html` : uClinux aux

pôles (NOAA, 2004)