Département Informatique INSA de Lyon

Pilote intégré de périphériques (VxWorks)

Jean-Philippe Babau

Département Informatique, laboratoire CITI
INSA Lyon

jean-philippe.babau@insa-lyon.fr

Département Informatique INSA de Lyon

jean-philippe.babau@insa-lyon.fr

Principes

- Accès à une ou plusieurs ressources (périphérique)
- Objectifs
  - masquer les contraintes matérielles
  - découpler l'application du matériel
  - protéger/partager l'accès
- Principes
  - un périphérique est vu comme un fichier
  - utilisation du système d'entrée/sortie (IOS)
  - primitives standardisées de l'IOS
    - appel par périphérique
    - open/creat/close/remove
    - read/write/ioctl
  - un périphérique est associé à un et un seul pilote
  - le pilote gère les requêtes sur le périphérique

jean-philippe.babau@insa-lyon.fr

Exemples

- Pilotes existants
  - gestion de l'accès à une liaison série
  - gestion de l'accès à une imprimante
  - gestionnaire de fichiers
- Liaison série
  - RS232 : voltage, CTS, RxD, TxD, adresse carte, …
  - Driver série

jean-philippe.babau@insa-lyon.fr

Exemple

```
char bufferE[20];  int fdE;

fdE=open("/pdev/ttyS0",O_WRONLY);
      {/*ouverture d'une liaison série en ecriture */
if (fdE < 0)
      {       printf(« erreur connexion\n");
              exit (-1);
       }
else
      {       bufferE [0] = 1   ; bufferE[1] = 0   ; bufferE[2] = 2  ;
              bufferE [3] = 65 ; bufferE[4] = 66 ; bufferE[5] = 13 ;
              write(fdE,bufferE,6);  /* envoi trame */
      } ;

close(fdE); /*arrêt connexion*/
```

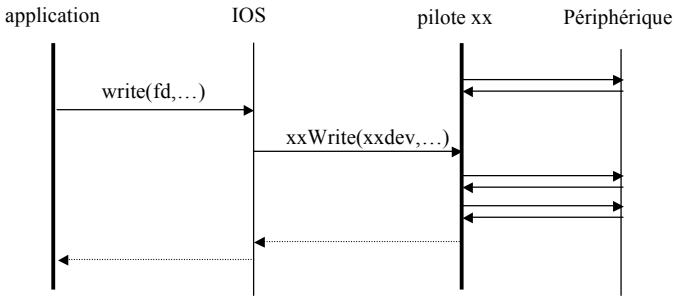
jean-philippe.babau@insa-lyon.fr

Exemple

```
char bufferR[20];  int fdR; int nCar; int nMax;

bufferR[0] = 0;
fdR=open("/pdev/ttyS1",O_RDONLY);              /*ouverture d'une liaison série en lecture*/
if (fdR < 0)
      {       printf(« erreur connexion\n");
              exit (-1);
       }
else
      {       while(bufferR[0]!= 1) {read(fdR, bufferR,1) ; }
              read(fdR, bufferR,1) ;
              if (bufferR[0]== 0)
              {               read(fdR, bufferR,1) ; nCar = buffeR[0];
                              nMax = read(fdR,bufferR,nCar) ; …
                              read(fdR, bufferR,1) ;
                              if (bufferR[0]== 13) {close(fdR);       /*arrêt connexion*/
                                                    exit(0) ;}
              }
      printf(« erreur communication");
      close(fdR);
      exit (-2);
       }
```

jean-philippe.babau@insa-lyon.fr

Principes (*VxWorks*)



- Appel de l'IOS synchrone ou asynchrone
- Logiciel dédié
  - lié au matériel, au SE

Logiciel à réaliser

jean-philippe.babau@insa-lyon.fr

L' I/O system (IOS)

- Service de l'OS
- Couche intermédiaire entre l'application et les pilotes
- Installations des services
  - installation / désinstallation d'un pilote
  - installation / désinstallation de périphériques (pilote identifié)
- Opérations sur les périphériques
  - connexion / lecture / écriture / configuration / déconnexion
  - L'appel du pilote est réalisé par l'IOS (**transparente pour l'utilisateur)**
    - recherche du pilote
    - appel de la primitive correspondante du pilote

jean-philippe.babau@insa-lyon.fr

Primitives de l'IOS pour un périphérique (*VxWorks*)

- Attacher
  - *fd = creat ("name ", flag) ;*
  - Connexion au périphérique *name*
  - fd : file descriptor (≠ 0,1,2) si connexion ok, ERROR sinon
  - flag : O_RDONLY, O_WRONLY, O_RDWR
- Ouvrir (idem attacher)
  - *fd = open ("name ", flag, mode) ;*
  - Connexion au périphérique *name*
  - fd : file descriptor (≠ 0,1,2) si connexion ok, ERROR sinon
  - flag : O_RDONLY, O_WRONLY, O_RDWR, O_CREAT, O_TRUNC
  - mode : 06444 sous unix
- Fermer (idem détacher)
  - *close (fd) ;*
  - Déconnexion du périphérique fd
- Détacher
  - *remove ("name ") ;*
  - Déconnexion du périphérique *name*

jean-philippe.babau@insa-lyon.fr

Département Informatique INSA de Lyon

Primitives de l'IOS pour un périphérique (*VxWorks*)

- Lire
  - *nBytes = read (fd,&buffer,maxBytes) ;*
  - Lecture de *maxBytes* caractères sur *fd* et stockage dans *buffer*
  - maxBytes : nombre maximum d'octets à lire
  - nBytes = nombre d'octets lus (-1 : erreur)
  - ERROR : non ouvert, pas de xxRead
- Ecrire
  - actualBytes = *write (fd,&buffer,maxBytes) ;*
  - Écriture des *maxBytes* caractères de *buffer* sur *fd*
  - maxBytes : nombre d'octets à écrire
  - actuelBytes : nombre d'octets écrits (si ≠ maxBytes : erreur)
  - ERROR : non ouvert, pas de xxWrite

jean-philippe.babau@insa-lyon.fr

Primitives de l'IOS pour un périphérique (*VxWorks*)

- Autre
  - *result = ioctl (fd, function,arg) ;*
  - Configuration de *fd* selon *function* et *arg*
  - function : code
  - arg : paramètre

  - exemple
    - *fd = creat(« COM1 »,* O_RDONLY*)*
    - *status = ioctl(fd,BAUDRATE,9600) ;*

Primitives asynchrones de l'IOS

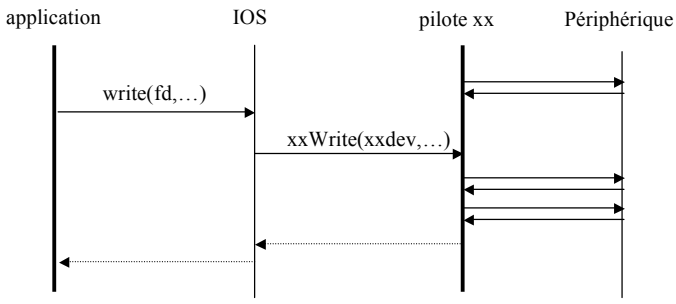
- Découplage application / pilote
  - Émission non bloquante d'une requête
- Création d'une structure par requête
  - Paramètres de l'appel (idem paramètres de l'appel synchrone)
  - Etat de la requête (en-cours, terminée, …)
  - Exemple pour un appel asynchrone de read :
    - char buffer[TAILLE] ;
    - aiocb_read.aio_fildes = fd ;
    - aiocb_read.aio_buf = buffer ;
    - aiocb_read.aio_nbytes = TAILLE ;
    - etc.

Primitives asynchrones de l'IOS

- Primitives
  - aio_read (& aiocb_read)
  - aio_write(& aiocb_write)
  - aio_return(& aiocb_write)
  - une utilisation simultanée d'une structure
- Suivi de la requête
  - (aio_error (& aiocb_read) = = EINPROGRESS)
  - aio_suspend(&aiocb[], nReq, timeout)
  - aio_cancel(&aiocb)
  - attente sur signal

jean-philippe.babau@insa-lyon.fr

Principes (*VxWorks*)



- Appel de l'IOS synchrone ou asynchrone
- Logiciel dédié
  - lié au matériel, au SE

Logiciel à réaliser

jean-philippe.babau@insa-lyon.fr

Primitives du pilote de gestion des périphériques

- Appelée par l'IOS
  - une par appel de l'IOS
- Paramètres
  - pointeur vers le descripteur du périphérique
  - paramètres de l'appel de l'IOS

```
int  xxCreat(XXDEV * desc, char * name, int  flag) ;

int  xxRemove(XXDEV * desc, char * name) ;

int  xxOpen(XXDEV * desc, char * name, int  flag) ;

int  xxClose(XXDEV * desc, char * name) ;

int xxRead (XXDEV * desc, char * buff, int  nBytes) ;

int xxWrite (XXDEV * desc, char * buff, int  nBytes) ;

int xxIOCtl (XXDEV * desc, int fonction, int  arg) ;
```

jean-philippe.babau@insa-lyon.fr

Primitives de l'IOS pour la gestion des pilotes

- Installation
  - Etablissement du lien primitives IOS / primitives du pilote
  - Mise en place d'un identifiant (numéro du Majeur)

```
drvNumber  = iosDrvInstall ( xxCreat, xxRemove, xxOpen, xxClose, xxRead, xxWrite, xxIOCtl)

drvNumber  = iosDrvInstall ( xxCreat, xxRemove, 0, 0, 0, xxWrite, xxIOCtl)
```

- Désinstallation

```
ret  = iosDrvRemove ( drvNumber,protOpen)
```

  - FALSE : pilote non désinstallé si un périphérique est ouvert
  - TRUE : un périphérique ouvert est fermé et détruit implicitement, le piloté est désinstallé

  - périphérique détruit implicitement
    - Pas d'appel à DevDel, mémoire perdue !

jean-philippe.babau@insa-lyon.fr

Primitives de l'IOS pour la gestion des périphériques

- Structure de description du périphérique
  - Identifiant
    - Nom
    - Numéro de mineur (Linux)
  - Lien avec le pilote
    - Numéro de Majeur
  - VxWorks : élément d'une liste : DEV_HDR (next, previous, N° de driver, nom)
  - infos spécifiques
    - adresses matérielles, données de configuraiton, …
  - mode R/W (non vérifié par IOS de VxWorks)
  - nombre maximum d'accès (")

- Ajout d'un périphérique

```
status  = iosDevAdd ( &desc, "name",drvNumber )
```

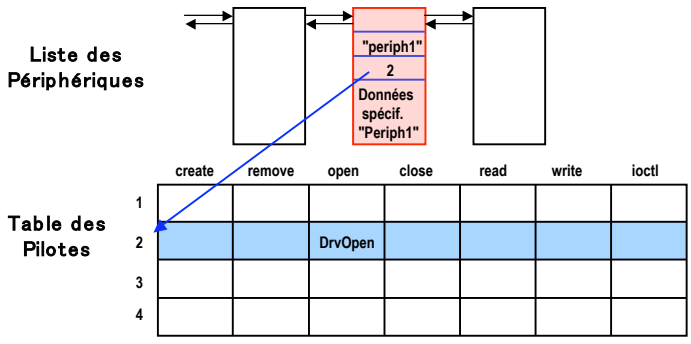
- Retrait d'un périphérique

```
status = iosDevDelete( pDevHdr );
```

jean-philippe.babau@insa-lyon.fr

Documentation VxWorks

- Structures de données du système d'E/S nécessaires pour la mise en œuvre d'un pilote
  - Une liste de périphériques, en général gérée dynamiquement
  - Une table des pilotes indexée par le majeur (le majeur désigne un pilote donné)



jean-philippe.babau@insa-lyon.fr

Documentation VxWorks

- Liste des périphériques
  - Structure de données du descripteur de périphérique
  - typedef struct

```
{
    DEV_HDR devHdr;
    XXspecific autres ;   <-- données spécifiques du périphérique
} XXDEV;
```

  - typedef struct

```
{
    DEV_HDR devHdr;
    int flag;
    char data[Taille_MAX];   } données spécifiques du périphérique
    …
} XXDEV;
```

jean-philippe.babau@insa-lyon.fr

Documentation VxWorks

- Installation d'un pilote "dev"

```
pilote_num = iosDrvInstall(&devCreate,0, &devOpen,0,&devRead,&devWrite,&devIoctl);
```

La fonction définit les routines du pilote pour les 7 fonctions d'E/S (1)

(4) Le système d'E/S renvoie le numéro de pilote Ex : pilote_num=2

Le système d'E/S détermine le prochain Slot libre dans la table des pilotes (2)

Le système d'E/S met à jour les routines du pilote dans la table (3)

Table des Pilotes

	xxCreate	xxRemove	xxOpen	xxClose	xxRead	xxWrite	xxIoctl
1							
2	&devCreate	0	&devOpen	0	&devRead	&devWrite	&devIoctl
3							
4							

jean-philippe.babau@insa-lyon.fr

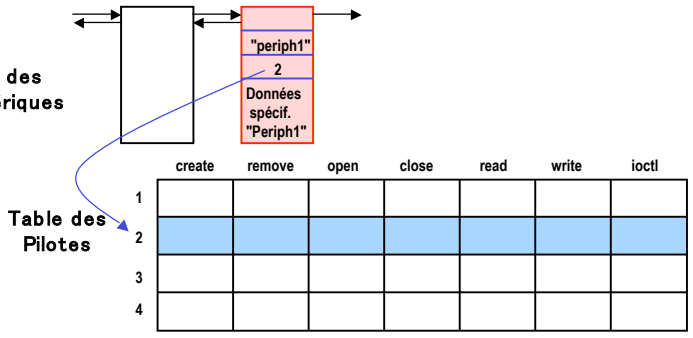
Documentation VxWorks

- Ajout de périphériques

```
status = iosDevAdd(&dev0," periph1", 2);
```

renvoie OK, ou ERROR si il existe un périphérique possédant le même nom

Le système d'E/S ajoute le descripteur de périphériques à la liste des périphériques



jean-philippe.babau@insa-lyon.fr

Documentation VxWorks

- Ouverture d'un périphérique

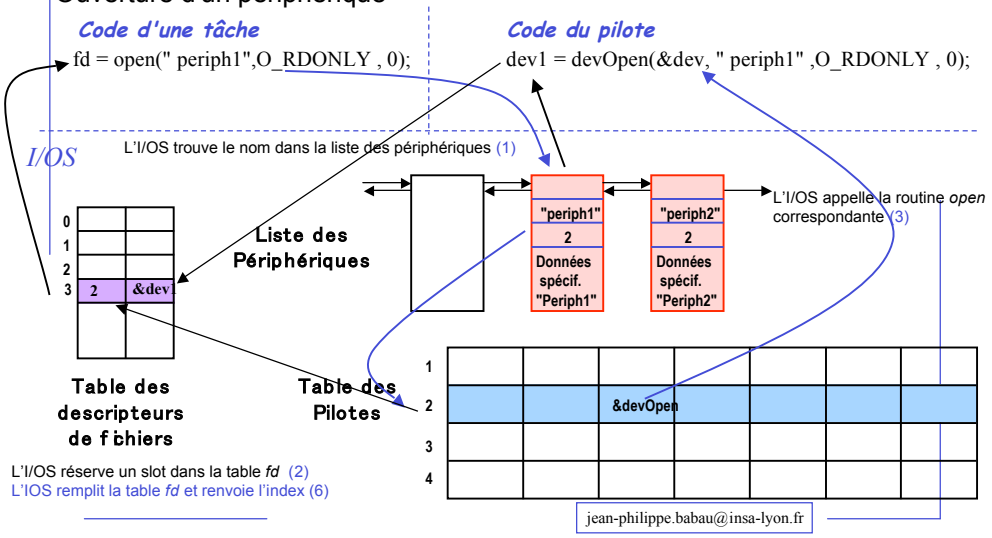
La primitive travaille sur les données spécifiques du périphérique (4)
Le pilote renvoie un pointeur sur le descripteur de périphérique (5)

Code d'une tâche

```
fd = open(" periph1",O_RDONLY , 0);
```

Code du pilote

```
dev1 = devOpen(&dev, " periph1" ,O_RDONLY , 0);
```



L'I/OS réserve un slot dans la table *fd* (2)
L'IOS remplit la table *fd* et renvoie l'index (6)

jean-philippe.babau@insa-lyon.fr

Documentation VxWorks

- Lecture sur un périphérique

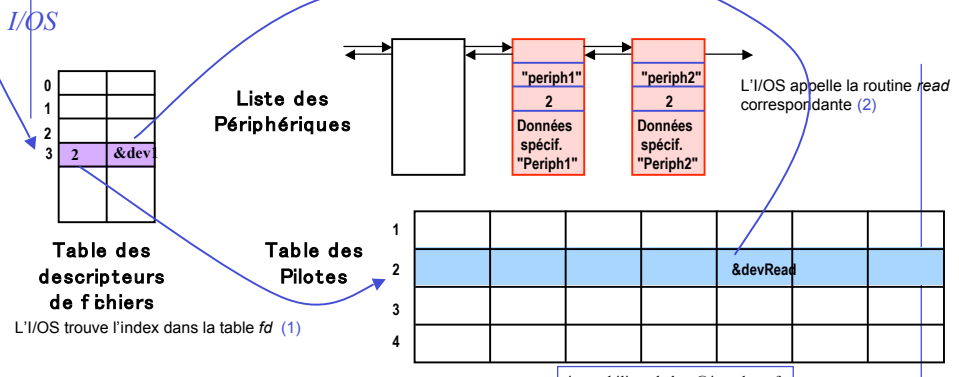
La primitive réalise l'opération de lecture (3)
Renvoie le nombre d'octets lus (4)

Code d'une tâche

nb = read(fd,tampon,longueur);

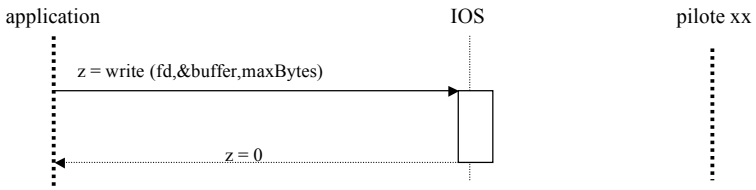
Code du pilote

nb = devRead(&dev1, tampon, longueur);

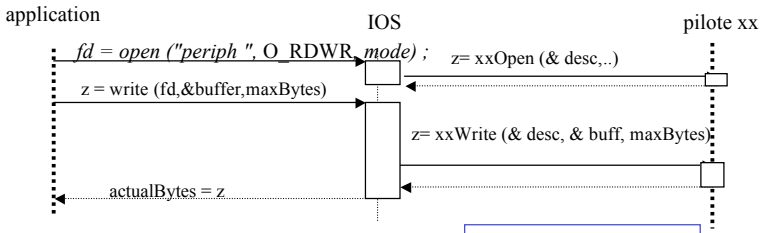


Comportement de l'IOS

Scénario : écriture sans ouverture préalable ou ouverture incorrecte



Scénario : écriture correcte avec ouverture préalable correcte

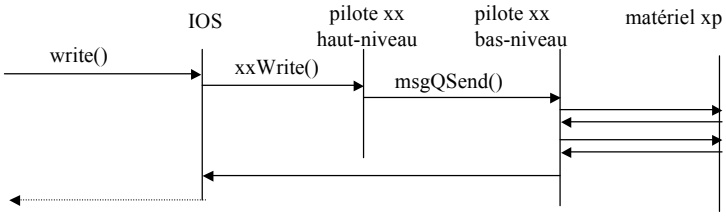


Vérifications

- IOS
  - nom du périphérique
  - connexion
    - pas de read avant open
  - nombre de connexions (non contrôlé par l'IOS de VxWorks)
  - type de connexion
    - pas de write sur O_RDONLY (non contrôlé par l'IOS de VxWorks)
- Driver
  - Ce qui n'est pas implémenté par l'IOS
  - Ce qui est spécifique au pilote
    - maxByte est trop important pour le périphérique concerné

jean-philippe.babau@insa-lyon.fr

Conception du pilote



- Partie haut niveau
  - primitives de l'IOS
- Découplage haut et bas niveau
  - boite aux lettres des requêtes
  - conservation de l'ordre d'appel
  - gestion des erreurs

jean-philippe.babau@insa-lyon.fr

Interface du pilote

- Primitives de gestion du pilote et déclaration de types

```
#ifndef PILOTE_H
#define PILOTE_H

#include <iosLib.h>            /* librairie pour les pilotes */

typedef struct                 /* structure d'échange des données périphériques : cast du buffer */
{       int valeur ;
} T_data ;

int DrvInstall();              /* primitive d'installation du pilote */
int DrvRemove();               /* primitive de désinstallation du pilote */

int DevAdd( char * name);      /* primitive d'ajout d'un périphérique */

int DevDel(char * name);       /* primitive de suppression d'un périphérique */

int  DrvConfig();              * primitive de configuration */
int  InfoDrv();                /* primitive de test */

#endif
```

jean-philippe.babau@insa-lyon.fr

Réalisation du pilote

Pilote.c

```
#include "Pilote.h"
#include <stdlib.h>

typedef struct          /* structure spécifique pour un périphérique */
{     int numero ;
} specific ;

typedef struct          /* structure standard  VxWoks pour un périphérique */
{     DEV_HDR donnees;
      specific autres;
} DEV;

/* Déclarations et implémentation des 7 primitives xxCreat, xxRemove, xxOpen, xxClose,
            xxRead, xxWrite, xxIOCtl */

/* Implémentation des primitives déclarées dans Pilote.h */
```

jean-philippe.babau@insa-lyon.fr

Primitives d'installation

```
int drvNumber = -1; int nbrePeriph = 0;

int DrvInstall()
{
      if (drvNumber==-1)
      {
      /* exemple d'installation de pilote */
      drvNumber  = iosDrvInstall ( &xxCreat, &xxRemove, &xxOpen, &xxClose, &xxRead, &xxWrite,
      &xxIOCtl) ;

      /* exemple d'installation de pilote */
      drvNumber  = iosDrvInstall ( NULL, NULL, &xxOpen, &xxClose,          NULL, &xxWrite,
      &xxIOCtl) ;
      }

      return drvNumber;
}
```

jean-philippe.babau@insa-lyon.fr

Département Informatique INSA de Lyon

Primitives d'installation

```
int DevAdd(char * name)
{
DEV * desc = (DEV *)malloc(sizeof(DEV));

      if (drvNumber != -1)
      {

      (desc->autres).numero = nbrePeriph ++  ;

      iosDevAdd ((DEV_HDR *)desc, name, drvNumber);

      return nbrePeriph ;
      }
else
      return -1;
}
```

jean-philippe.babau@insa-lyon.fr

Primitives de désinstallation

```
int DevDel(char * name)
{
    DEV_HDR * pDevHdr;      /* structure pour le périphérique */
    char* suite [1];

    pDevHdr = iosDevFind(name,suite);/* recherche du périphérique */
    if ((pDevHdr!= NULL) &&  (*suite[0]==‘\0’))
    {
        iosDevDelete( pDevHdr );
        free(pDevHdr);
    }
    return 0;
}

int DrvRemove()
{   int ret;/* numéro du pilote */
    if (drvNumber != -1)
    {       ret  = iosDrvRemove( drvNumber,FALSE) ;
        drvNumber = -1 ;}
    return ret ;
}
```

jean-philippe.babau@insa-lyon.fr

Primitives liés au périphérique

```
int  xxOpen(DEV * desc, char * remainder, int  mode)
{
    if (*remainder!=‘\0’))
    {
        return ERROR;
    }
    else
    {
        return ((int) desc) ;
    }
}

int  xxClose(DEV * desc, char * name)
{
    return 0;
}
```

jean-philippe.babau@insa-lyon.fr

Primitives liés au périphérique

```
int xxWrite (DEV * desc, char * buff, int  nBytes)
{
    /* pas de printf dans un driver !! */
    printf("periph %d : %s \n",    (desc->autres).numero , buff);
    return 0;
}

int xxIOCtl (DEV * desc, int fonction, int  arg)
{
    if (fonction= =1)
    {
        (desc->autres). numero = - (desc->autres). numero ;
    }
    return 0;
}
```

jean-philippe.babau@insa-lyon.fr

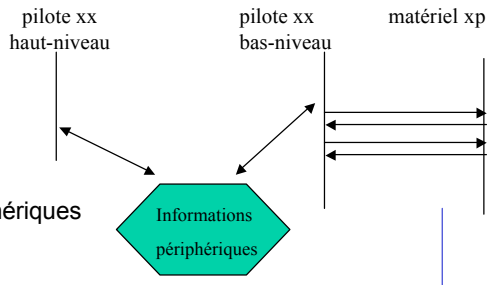
Appel du pilote

```
#include " Pilote.h"

int main(void)
{
int numPilote;
int fd1;
int fd2;

numPilote = DrvInstall();

DevAdd( "periph1");
DevAdd( "periph2");

fd1=open("periph1",O_RDWR,0);
fd2=open("periph2",O_RDWR,0);

write(fd1,"Hello",6) ;
write(fd2," world",7) ;

ioctl(fd1,0,0);
ioctl(fd2,1,0);

write(fd2," world",7) ;

close (fd2);
close (fd1);

DrvRemove();

return (0);}
```

jean-philippe.babau@insa-lyon.fr

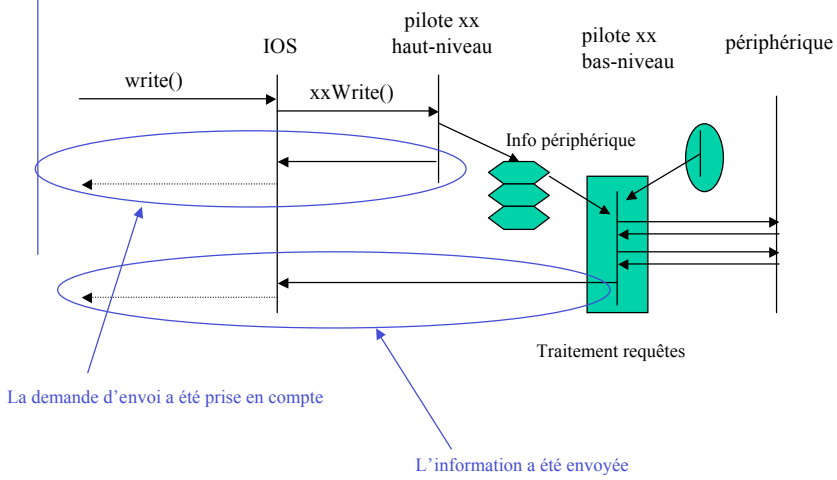
Informations gérées par le pilote

- Vue du périphérique
  - État du périphérique
  - Informations reçues
  - Informations datées
- Informations à destination du périphériques
  - Données à mémoriser
  - Données à afficher, imprimer, …
  - Commandes à réaliser
- Le pilote gère une vue du périphérique
  - Initialisation (adresse, …) : DevAdd
  - Configuration : ioctl
  - Connexion : open/close ou creat/remove
  - Lecture / écriture : read / write



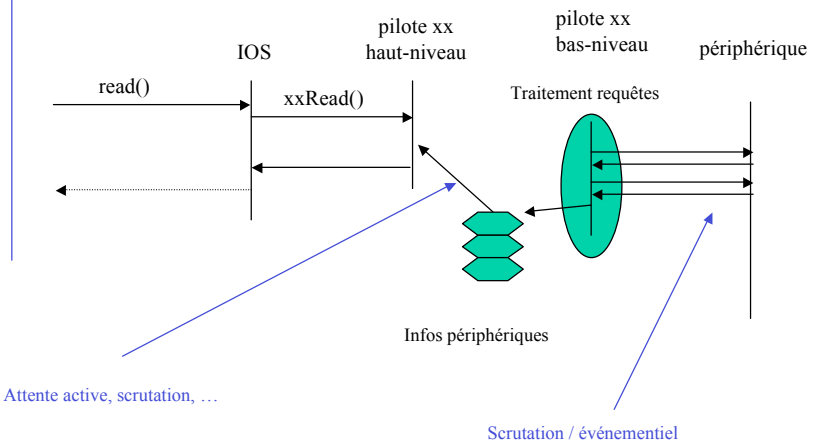
jean-philippe.babau@insa-lyon.fr

Ecriture



jean-philippe.babau@insa-lyon.fr

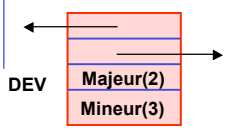
Lecture



jean-philippe.babau@insa-lyon.fr

Mise en place des informations des périphériques

- Données de bas niveau
  - Registres, ports
- Structure logicielle de stockage
  - Tableau(mineur), *liste (à éviter)*
- Lien primitives haut niveau/ bas niveau (mineur, DEV.specific)



Indice(mineur)	0	1	2	**3**										15
isAdded	0	1	0	**1**										0
Adresse HW	-1	23	-1	**24**										-1
Infos spécifiques														

Tableau des informations

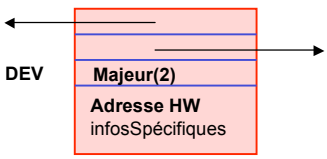


Tableau ou liste des DEV&

jean-philippe.babau@insa-lyon.fr

Architecture du pilote

- Lien avec le périphérique réel
  - Communication avec un composant matériel
  - Dépendant du matériel : IT, polling
- Primitives de haut niveau
  - Traitement direct ou appel d'un serveur (tâche)
  - 1 tâche par requête / 1 tâche pour l'ensemble des requêtes
- Modèle de communication haut niveau et bas niveau
  - Politiques de consommation, de mémorisation, de perte
  - Zone statique / dynamique
  - Aspects temporels
- Cf. cours conception multitâches (concurrence, échange de données)

Impact sur la spécification des primitives de communication

jean-philippe.babau@insa-lyon.fr

Gestion des éléments

- Création/initialisation, destruction
  - statique
    - lors de l'appel de *DrvInstall*, *DrvDesinstall*
  - dynamique
    - appels spécifiques de *xxIOctl*
    - premier *xxCreat* / dernier *xxRemove*
- Gestion des erreurs
  - pilote inexistant, déjà installé
  - périphérique déjà créé, …

jean-philippe.babau@insa-lyon.fr

Gestion des éléments

- Configuration du pilote
  - Install, primitive dédiée
  - le pilote est un périphérique spécial
- Configuration des périphériques
  - appel de DevAdd
  - appels spécifiques de *ioctl*
    - liaison série

```
status = ioctl(fd,BAUDRATE,9600) ;
```

jean-philippe.babau@insa-lyon.fr

Architecture en couches

- Fichier sur un disque
  - pilote de disque
  - pilote de fichiers
- Lecteur accessible via une liaison série
  - pilote de liaison série
  - pilote du lecteur
- Imprimantes sur un réseau
  - pilote du réseau
  - pilote de l'imprimante
- Périphérique en réseau
  - pilote du réseau
  - pilote de messages
  - pilote de périphériques

jean-philippe.babau@insa-lyon.fr

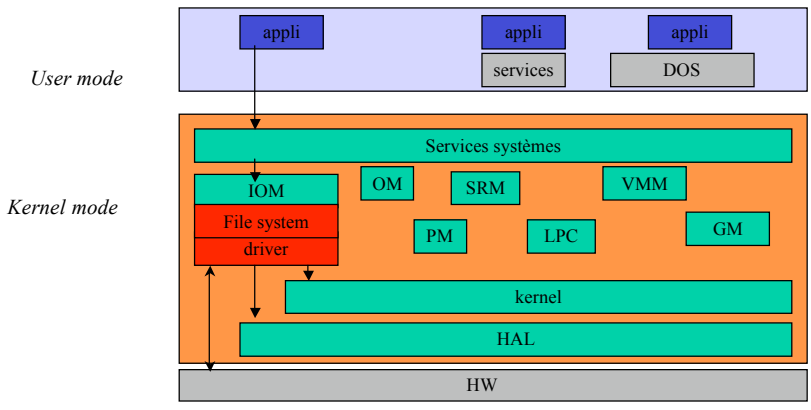
OS classiques

- Séparation application / noyau
  - Droits et accès
    - Application : adressage virtuel et pas d'accès aux IT
    - Noyau : adressage réel et accès aux IT
  - Les primitives noyaux sont spécifiques
    - Protection du système
  - Communication de données Application / Noyau
    - Mécanismes spécifiques de copie
    - Linux : *copy_to_user*, *copy_from_user*
- Driver : espace noyau
- Gestion du bas niveau
  - Gestion des IT
    - Windows : ISR -> DPC
  - Gestion DMA

jean-philippe.babau@insa-lyon.fr

Architectures des systèmes

- OS généraliste (NT4)

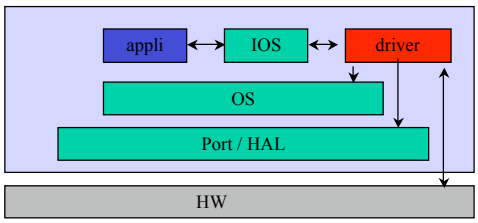


jean-philippe.babau@insa-lyon.fr

Architectures des systèmes

- RTOS

Mode *user*

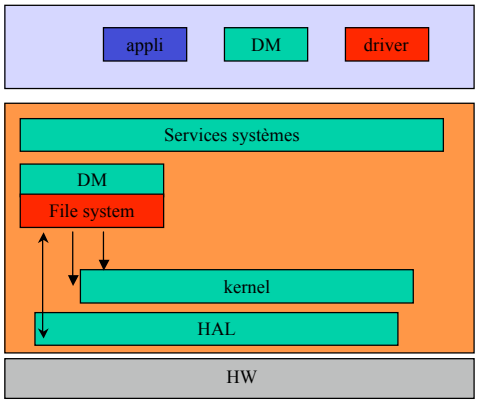


Architectures des systèmes

- Windows CE

Mode *user*

Kernel mode



Windows

• I/O manager : IOS

• Primitives de l'I/O manager
CreateFile(), WriteFile(), ReadFile(), DeviceIoControl(), CloseHandle()

• IRP : structure de données contenant toutes les informations nécessaires au traitement d'une requête. Par exemple, pour une requête de lecture dans un fichier, l'IRP va contenir un buffer, la longueur des données à lire, etc.

jean-philippe.babau@insa-lyon.fr

Windows

– lancement du driver
DriverEntry(PDRIVER_OBJECT *DriverObject*, PUNICODE_STRING *RegistryPath*)

- Primitives définies dans DRIVER_OBJECT des primitives, du driver

– ajout d'un périphérique
*xx*AddDevice(PDRIVER_OBJECT*DriverObject*,PDEVICE_OBJECT*PhysicalDeviceObject*)

- requête d'E/S
XXDispatchRead(PDEVICE_OBJECT DeviceObject,PIRP Irp);
XXDispatchWrite, XXDispatchCreate, XXDispatchClose, XXDispatchDeviceControl

– arrêt du pilote
XXUnload(PDRIVER_OBJECT DriverObject)

– arrêt d'une requête
Cancel(PDEVICE_OBJECT *DeviceObject*,PIRP *Irp*)

jean-philippe.babau@insa-lyon.fr

Windows

- Fonctionnement

1- La primitive Windows ReadFile() est appelée par l'application en mode Utilisateur.

2 - Si la demande est valide, elle est transformée en IRP, puis est transmise au bon pilote en lui indiquant le périphérique concerné.

3 - Les IRP sont traitées par la routine spécifique du driver (XXDispatchYY)

4 - l'IRP est complétée avec la réponse à la requête, le I/O Manager est avisé de la réponse.

5 - L'IRP subit la transformation inverse par le I/O Manager et la réponse est renvoyée à l'application.

jean-philippe.babau@insa-lyon.fr

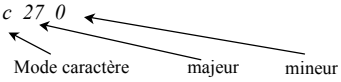
Linux

- Pilote
  - numéro majeur *cat /proc/devices*
    - Système ouvert : utilisation indépendante de l'installation, majeur prédéfini
      - Assignés : 0-230; non assignés 231-239; local use : 240-254
    - Assignation dynamique possible
  - primitives pour l'installation x*init_module, xcleanup_module*
    - Association primitives IOS / primitives pilote
- Installation
  - Chargement de module noyau
    - `insmod Pilote.ko`
    - `appel de init_module`
  - Déchargement de module noyau
    - *rmmod* `Pilote`
    - `Appel de cleanup_module`

jean-philippe.babau@insa-lyon.fr

Linux

- Périphérique
  - Fichier
    - Identifié par un nom de fichier, un numéro de majeur et un numéro de mineur
  - Ajout d'un périphérique
    - *mknod /dev/ periph c 27 0*



  - *"/dev/periph"*
    - mode (caractère/bloc) numéro majeur numéro mineur
    - file /dev/periph
  - Retrait d'un périphérique
    - *rm /dev/ periph*
- Primitives de l'IOS
  - open, read, write, close, ioctl

Linux

```
#define XX_NAME "XX"
int major  = 27 ;
…
struct file_operations XX_fops =
{
.owner =THIS_MODULE,
                        /* lseek,*/
.read = XX_read,         /* read */
                        /* write */
                        /* readdir */
                        /* poll */
.ioctl = XX_ioctl,      /* ioctl */
                        /* mmap */
.open = XX_open,        /* open*/
                        /* flush */
.release =  XX_release  /* release*/
};
```

```
int xinit_module(void)
{ …
 if(register_chrdev(major, XX_NAME, &XX_fops) < 0)
/* si major = 0 : demande de numero de majeur */
        {
 printk(" cannot allocate major number, not loaded.\n");
        } …
}
void xcleanup_module(void)
{
…
if(unregister_chrdev(major, XX_NAME)) {
   printk("unregister_chrdev() failed\n");
  }
}

      /* primitives noyaux : printf -> printk */
```

Pilotes fournis par les OS

- Pilotes existants
  - pilote terminal
  - pilote de RAM
  - pilote de fichiers
  - pilote de pipe
  - pilote NFS
- Pilote terminal
  - tyDevInit()
  - paramétrisation via ioctl
  - fonctions
    - FIOSETOPTIONS, OPT_LINE (arrêt sur NEWLINE)
    - FIOBAUDRATE, vitesse
    - FIOFLUSH, 0
    - FIOCANCEL (arrêt d'un read/write), 0

jean-philippe.babau@insa-lyon.fr

Exemple : pipe sous VxWorks

```
char string[20];  int fd;

if((fd=open("/pipe/MBePipe",O_RDWR,0))==ERROR)
      {/*creation d un pipe recevant des messages de 5 caracteres et pouvant  en contenir 4 max puis ouverture en
      ecriture et lecture*/  /* pipeDevCreate : appel de iosDevAdd */

      if(pipeDevCreate("/pipe/MBePipe",4,5)==ERROR || (fd=open("/pipe/MBePipe",O_RDWR,0))==ERROR)
            {        printf("Echec a la creation / ouverture de MBePipe\n"); }
      printf("proc1 : Creation et ouverture\n"); }
else
      {     printf("proc1 : Ouverture\n"); }
do
      {     scanf("%20s",string);
            printf("resultat d ecriture : %i\n",write(fd,string,3));
            /*Meme en ecrivant que 3 caracteres a chaque fois le pipe est plein apres
            4 ecritures si aucune extraction n a lieu entre temps*/
      }while (string[0]!='f' || string[1]!='i' || string[2]!='n');

close(fd);     /*fermeture mais pas destruction*/
```

jean-philippe.babau@insa-lyon.fr

Exemple : pipe sous VxWorks

```
int fd;
char string[10];
if((fd=open("/pipe/MBePipe",O_RDWR,0))==ERROR)
      {
      if(pipeDevCreate("/pipe/MBePipe",4,5)==ERROR || (fd=open("/pipe/MBePipe",O_RDWR,0))==ERROR)
            {
            printf(" Echec a la creation/ouverture de MBePipe\n");
            }
            printf(" Creation et ouverture\n");
      }
else
      {
      printf(" Ouverture\n");
      }
do
      {
      read(fd,string,5);
      printf("proc2 : %s\n",string);
      }while(string[0]!='f' || string[1]!='i' || string[2]!='n');

close(fd);
```

jean-philippe.babau@insa-lyon.fr

Les périphériques virtuels

- Définition de périphériques virtuels
  - Élément de l' IHM (souris, clavier, écran)
  - VirtualMouse de Windows
  - Périphérique par défaut
- Pilote de périphériques virtuels
  - Livré avec l'OS
  - Couche service adaptée
    - onClick()
  - Primitives du pilote inaccessibles
  - ? Pilote ?
- Pilote réel
  - Réalisation du pilote bas niveau
  - Interconnexion avec le périphérique virtuel
  - Émulation du périphérique virtuel

jean-philippe.babau@insa-lyon.fr

Périphérique virtuel

- Inconvénients
  - Limitation du comportement
    - Pas de triple click
    - Comportement par défaut
  - QoS
- Avantages
  - Développement rapide de pilotes
  - Applications indépendantes des périphériques
  - Fiabilité et sécurité des noyaux

jean-philippe.babau@insa-lyon.fr

Périphériques « distants »

- Réseau de périphériques
  - Réseaux de capteurs, réseaux d'appareils, périphériques sans fil
- Profils de haut niveau intégrés dans les protocoles
  - Surcouche aux protocoles de transport
  - Périphériques virtuels
  - Spécification génériques
    - Nom, adresse, liste d'opérations ou de services
- Profils Bluetooth
  - Fax Profile (FAX) : profil de télécopieur
  - Headset Profile (HSP) : profil d'oreillette
  - Serial Port Profile (SPP) : profil de port série
- Le protocole avec ses profils remplace le driver
  - Échanges standardisés
  - Communication (appareil photo, téléphone, imprimante) <-> ordinateur

jean-philippe.babau@insa-lyon.fr

La couche service

- Interconnection application / pilote
  - mise en forme des données
    int GetSpeed(int v)
    { … read (fdABS,&buffer,maxBytes) ; …}
  - protocole de communication
- Administration, contrôle, protections
  - administrateur : installation/libération/réinitialisation
  - Utilisateur
    - droit et temps d'accès
    - partage

jean-philippe.babau@insa-lyon.fr

La couche service

- Fonctions avancées
  - accès multiples
    - creat ( ); creat ( );
  - envois multiples
    - write( ); write( )
  - envois avec attente
    - taskDelay ( ); write()
  - échanges entre périphérique
    - read( ); write() /* échanger */

jean-philippe.babau@insa-lyon.fr

Exemple de couche service : le joystick sous Windows

UINT joyGetNumDevs(VOID)

The joyGetNumDevs function returns the number of joysticks supported by the current driver or zero if no driver is installed.

MMRESULT joyGetPos(UINT *uJoyID*, LPJOYINFO *pji*)

uJoyID Identifier of the joystick to be queried. Valid values for *uJoyID* range from zero (JOYSTICKID1) to 15

pji Pointer to a JOYINFO structure that contains the position and button status of the joystick.

Returns JOYERR_NOERROR if successful or one of the following error values.

jean-philippe.babau@insa-lyon.fr

Exemple de couche service : le joystick sous Windows

MMRESULT joySetCapture(HWND *hwnd*, UINT *uJoyID*, UINT *uPeriod*, BOOL *fChanged*)

hwnd Handle to the window to receive the joystick messages.

uJoyID Identifier of the joystick to be captured. Valid values for *uJoyID* range from zero (JOYSTICKID1) to 15

uPeriod Polling frequency, in milliseconds.

fChanged Change position flag. Specify TRUE for this parameter to send messages only when the position changes by a value greater than the joystick movement threshold. Otherwise, messages are sent at the polling frequency specified in *uPeriod*.

Returns JOYERR_NOERROR if successful or one of the following error values.

jean-philippe.babau@insa-lyon.fr

Conclusion

- Un pilote gère des périphériques tels des fichiers
- Un périphérique est géré par un seul pilote
- IOS : couche intermédiaire standardisée application/pilotes
- Pilote
  - Architecture dédiée : OS / matériel
  - Gestion d'une vue du périphérique
    - Correspondance avec les actions sur le périphérique
  - Architecture logicielle
    - Politique de stockage/consommation des informations
    - Concurrence
    - Statique / dynamique
    - Architecture en couches
- Couche service : services de haut niveau

jean-philippe.babau@insa-lyon.fr