Administration système

Noyau

Tuyêt Trâm DANG NGOC
<dntt@u-cergy.fr>

Université de Cergy-Pontoise

2009–2010

Système UNIX

- un noyau
- des bibliothèques, logiciels, fichiers, etc.

Noyau

Le noyau assure la gestion des ressources physiques et logiques du système :

- gestion des périphériques par les pilotes intégrés au noyau
- gestion des fichiers associée à la gestion des périphériques
- gestion des processus à l'aide de l'ordonnanceur (« process scheduler »

Un noyau UNIX est un fichier exécutable qui se trouve sur le disque de boot.

- Cœur du système UNIX
- Programme lancé au démarrage
- Programme simple où presque tous les paramètres sont fixés statiquement
- Les systèmes récents ont quelques paramètres dynamiques

Pourquoi configurer (et compiler) un noyau ?

Les noyaux disponibles sont souvent compilés avec un maximum de pilotes, dans le but d'être compatibles avec une majorité de machines.

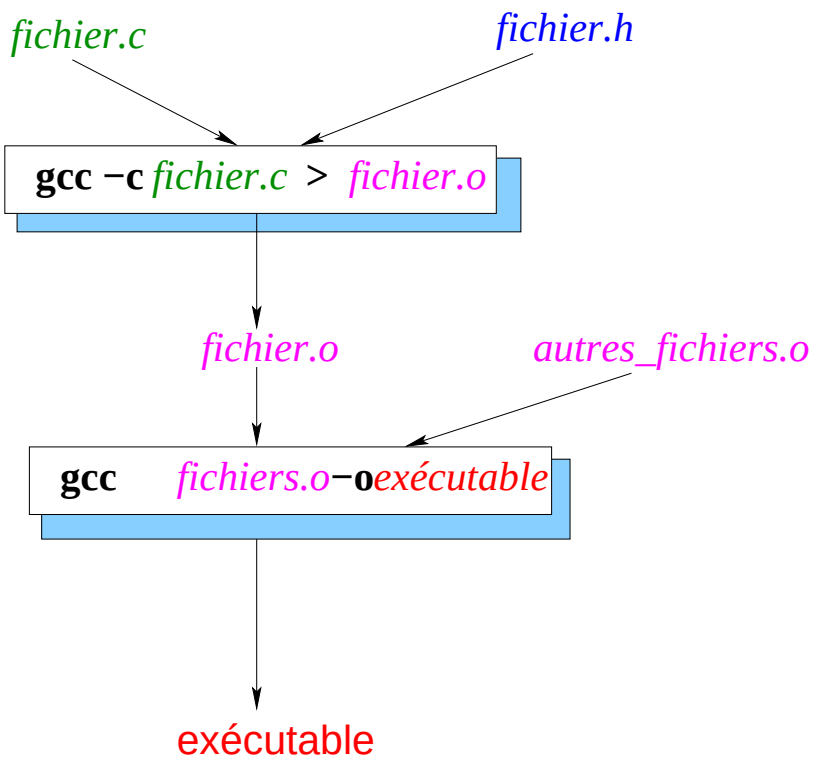
- ajout de périphériques
- suppression de pilotes inutiles
- tuning : paramétrage pour l'optimisation ou des besoins spécifiques,
- application de correctifs (patch)

Pourquoi mettre un jour un noyau ?

- vous avez un matériel dont le support a été ajouté dans une nouvelle version du noyau
- un trou de sécurité a été découvert dans le noyau actuel, ce problème étant réglé dans une nouvelle version
- vous souhaitez toujours avoir le dernier noyau possible
- *comme tout bon Geek, vous aimez compiler votre kernel :-)*

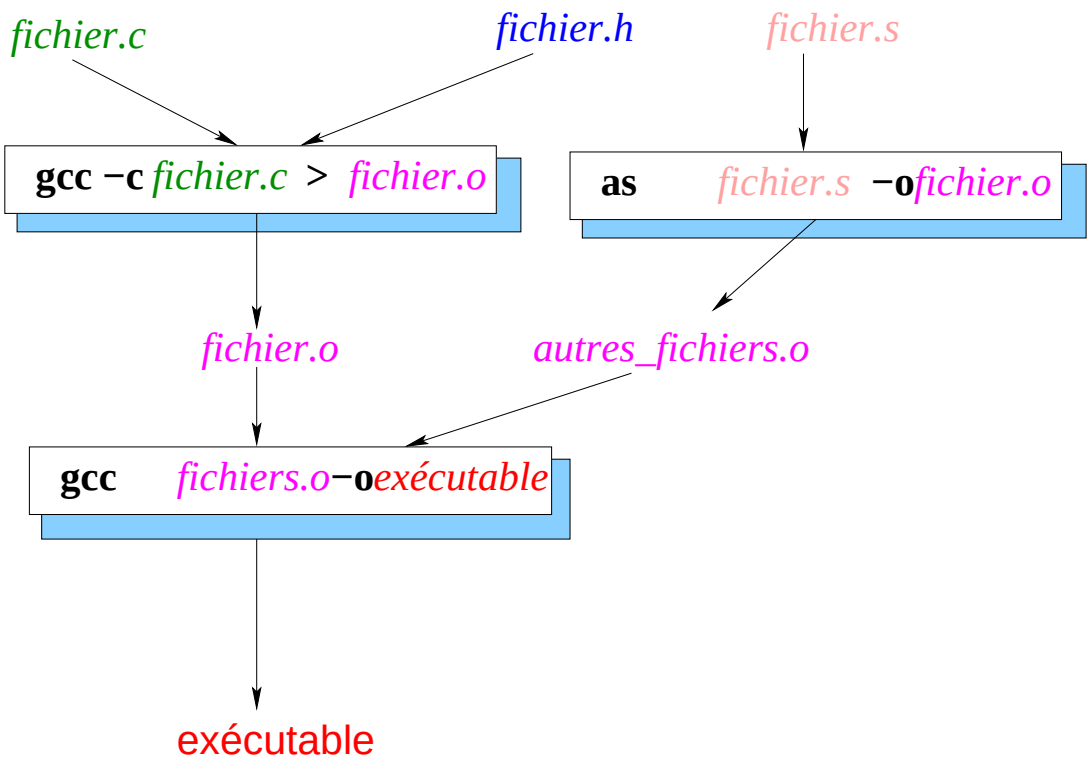
Comment reconfigurer un nouveau noyau ?

Noyau : binaire exécutable écrit en C avec un peu d'assembleur.
Les sources se compilent comme n'importe quel logiciel écrit en C.



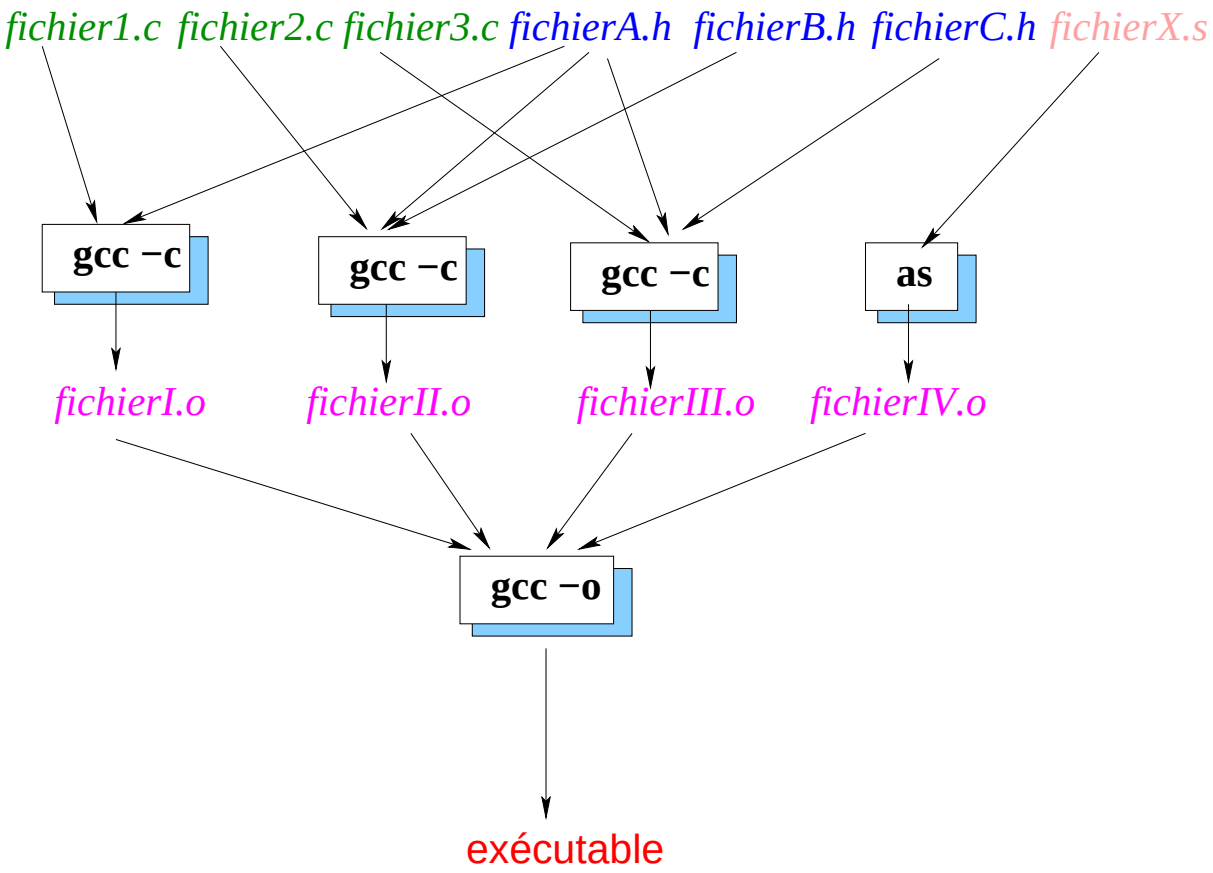
Comment reconfigurer un nouveau noyau ?

Noyau : binaire exécutable écrit en C avec un peu d'assembleur.
Les sources se compilent comme n'importe quel logiciel écrit en C.



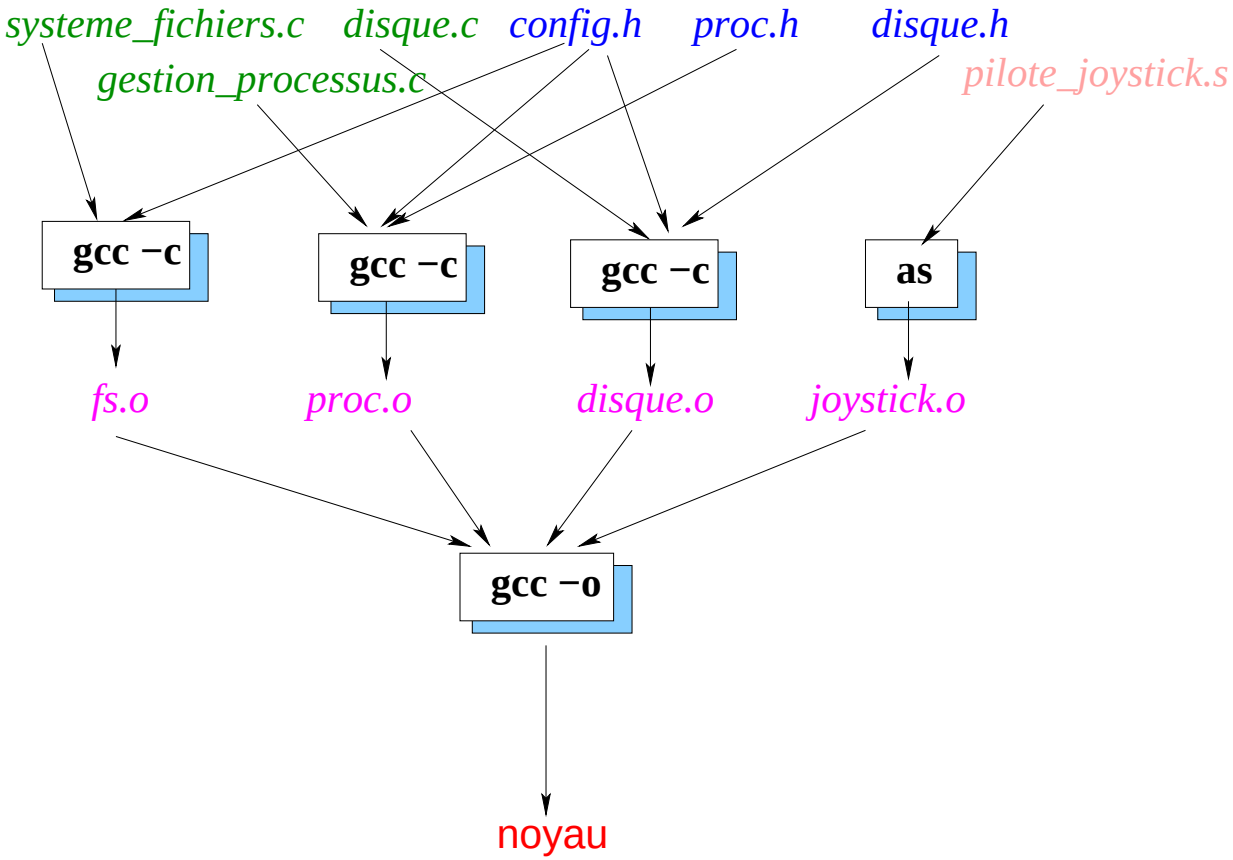
Comment reconfigurer un nouveau noyau ?

Noyau : binaire exécutable écrit en C avec un peu d'assembleur.
Les sources se compilent comme n'importe quel logiciel écrit en C.



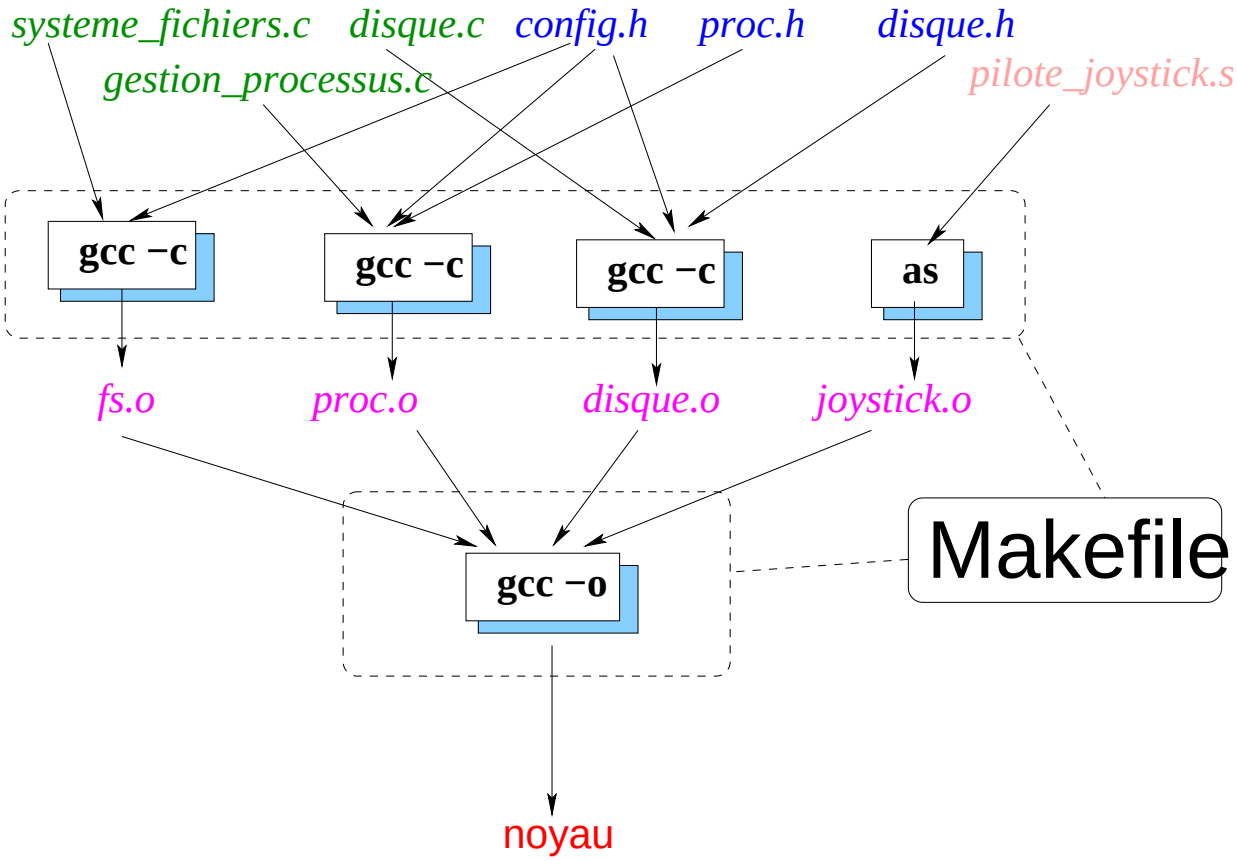
Comment reconfigurer un nouveau noyau ?

Noyau : binaire exécutable écrit en C avec un peu d'assembleur.
Les sources se compilent comme n'importe quel logiciel écrit en C.

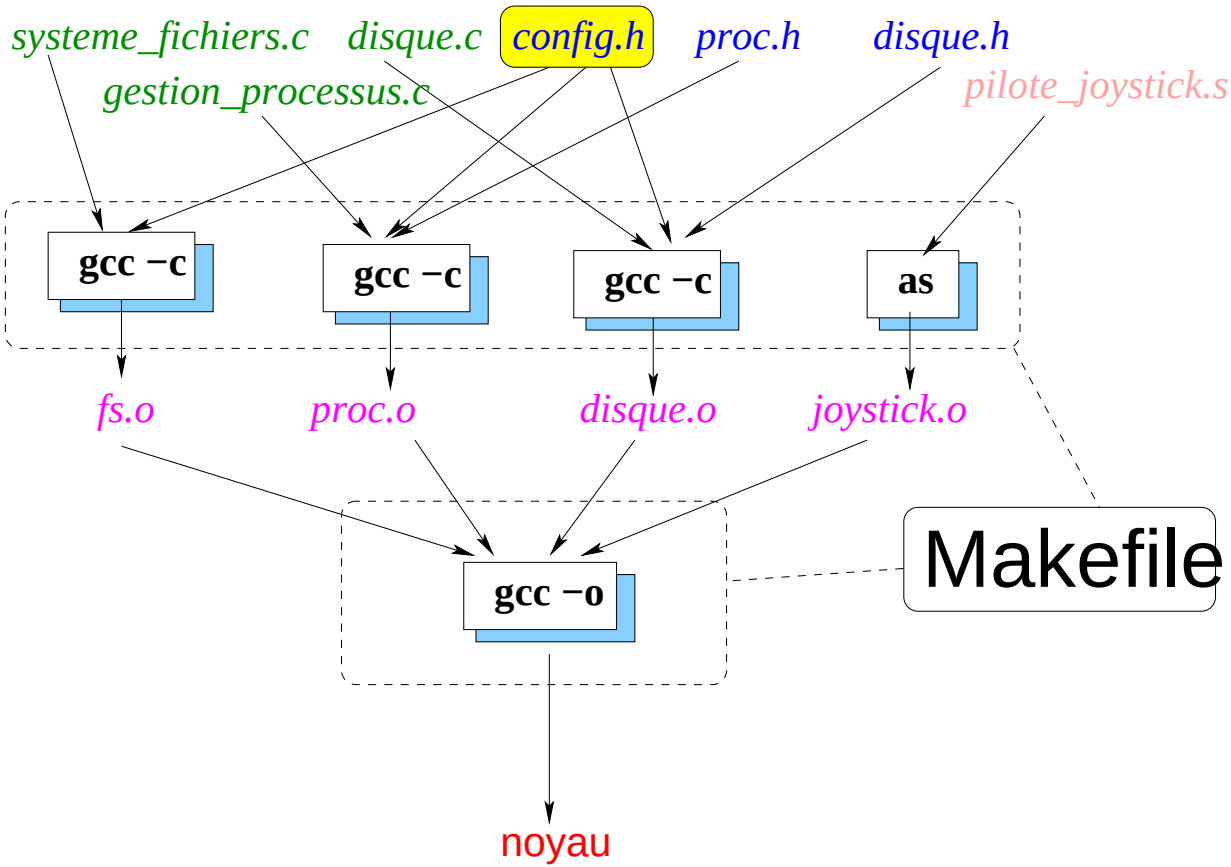


Comment reconfigurer un nouveau noyau ?

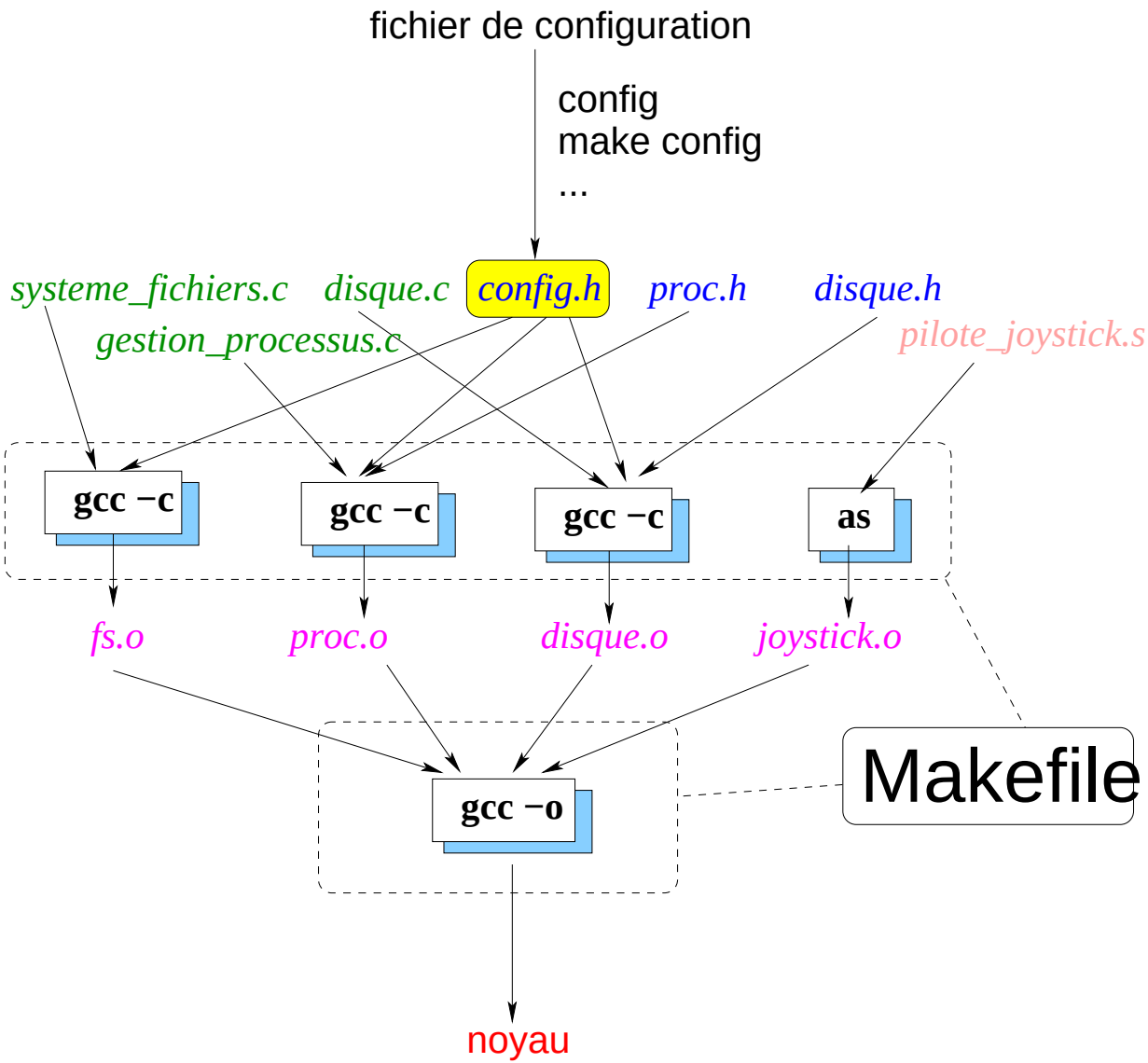
Noyau : binaire exécutable écrit en C avec un peu d'assembleur.
Les sources se compilent comme n'importe quel logiciel écrit en C.



Fichier de configuration



Fichier de configuration



Comment installer un nouveau noyau ?

1. en général, un fichier de configuration qui est lu pour générer les `.h` nécessaires pour générer les noyau
   1. éditer ce fichier de configuration suivant la syntaxe demandé. Sur certains systèmes, une interface plus ou moins conviviale permet de générer ce fichier.
   2. générer les fichiers .h correspondant à cette configuration
2. compiler en utilisant le `Makefile` donné.
3. l'exécutable est alors généré, c'est le **noyau**. Suivant les systèmes, il peut se nommer `kernel`, `zImage`, `bzImage`, `vmunix`, etc.
4. installer le nouveau noyau à la place de l'ancien
   1. sauvegarder l'ancien
   2. copier le nouveau à la place de l'ancien

Avant de configurer son noyau

- Bien connaître son matériel (`dmesg`, `lspci`, `lsusb`, `lshw`, `hwinfo`, `hdparm`, `cat /proc/cpuinfo /proc/acpi/info`, etc.)
- Bien connaître les services dont on a besoin (firewall, dimensionnement pour BD, IPv6, etc.)

Configuration d'un noyau FreeBSD

- Fichier de configuration à éditer
  - description du matériel
  - sélection d'options
- Création d'une nouvelle configuration :

```
config fichier_configuration
```

- Compilation du noyau :

```
make depend
make
```

- Installation

```
make install
make module-install
```

- Redémarrage

Fichier de configuration BSD

- Suite de déclarations :
  - `machine type`
  - `cpu type`
  - `ident nom_noyau`
  - `maxusers nombre_d_utilisateurs`
  - `options option`
  - `config racine_et_swap`
  - `controller`
  - `disk`
  - `tape`
  - `device`
  - `pseudo-device`

Exemples de déclarations (1)

```
machine         "i386"
cpu             "I686_CPU"
ident           PCCHEF
maxusers        128
options         INET                    # Support réseau IP
options         FFS                     # Berkeley Fast Filesystem
options         FFS_ROOT                # FFS usable as root device [ke
options         "COMPAT_43"             # Compatible with BSD 4.3 [KEEP
config          kernel  root on ad0 # où se trouvera le noyau
```

Exemples de déclarations (2)

```
controller      isa0
controller      pci0
controller      fdc0    at isa? port "IO_FD1" bio irq 6 drq 2
disk            fd0     at fdc0 drive 0
disk            fd1     at fdc0 drive 1
controller      wdc1    at isa? port "IO_WD2" bio irq 15
disk            wd2     at wdc1 drive 0
disk            wd3     at wdc1 drive 1
options         ATAPI           #Enable ATAPI support for IDE bus
options         ATAPI_STATIC    #Don't do it as an LKM
device          acd0            #IDE CD-ROM
```

Fichier de configuration FreeBSD

Par défaut dans `/usr/src/sys/i386/conf`, on trouve :

- `LINT` qui est une config qui contient tous les paramétrages et options possibles
- `GENERIC` qui est un config qui contient les paramétrages supportant la plupart des configuration matérielle standard et répondant à la plupart des besoins standard. C'est ce fichier de configuration qui a été utilisé pour générer le noyau par défaut sur le système.

Pour créer un nouveau noyau :

- **cp GENERIC MON_NOYAU**
- éditer `MON_NOYAU` en :
  - supprimant les options inutiles
  - adaptant les paramètres
  - ajoutant de nouveaux paramètres, périphériques, support en s'inspirant du fichier `LINT`

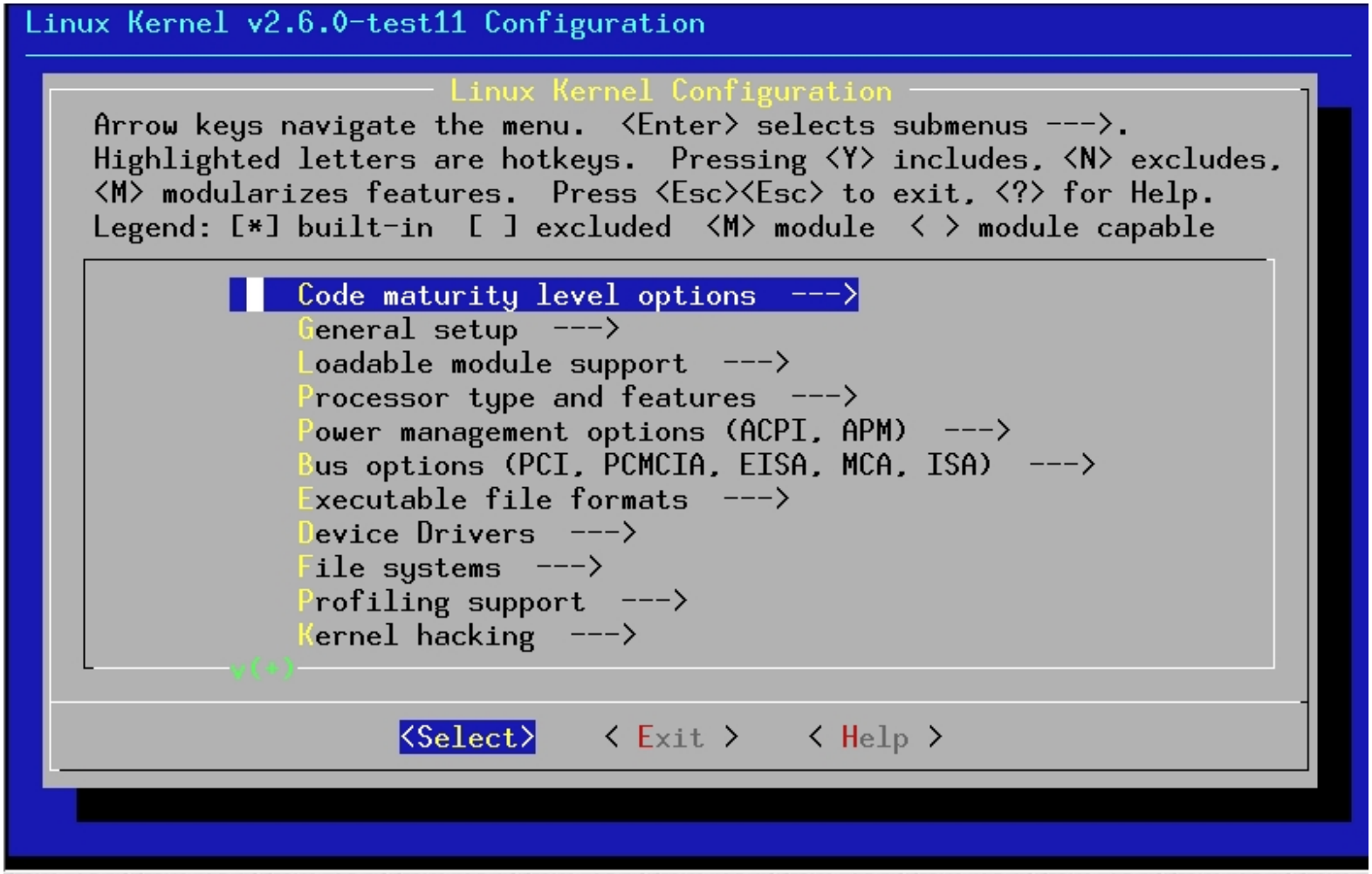
Configuration d'un noyau Linux

- Configuration, dans le répertoire `/usr/src/linux`
  - Edition du fichier `/usr/src/linux/.config`
  - ou `make config` (question-réponse)
  - ou `make menuconfig` (interface ncurses)
  - ou `make xconfig` (interface graphique)
  - ou `make gconfig` (jolie interface graphique)
- Compilation du noyau :
  - `make depend`
  - `make zImage` ou `make bzImage`
  - `make modules`
- Installation
  - `make install`

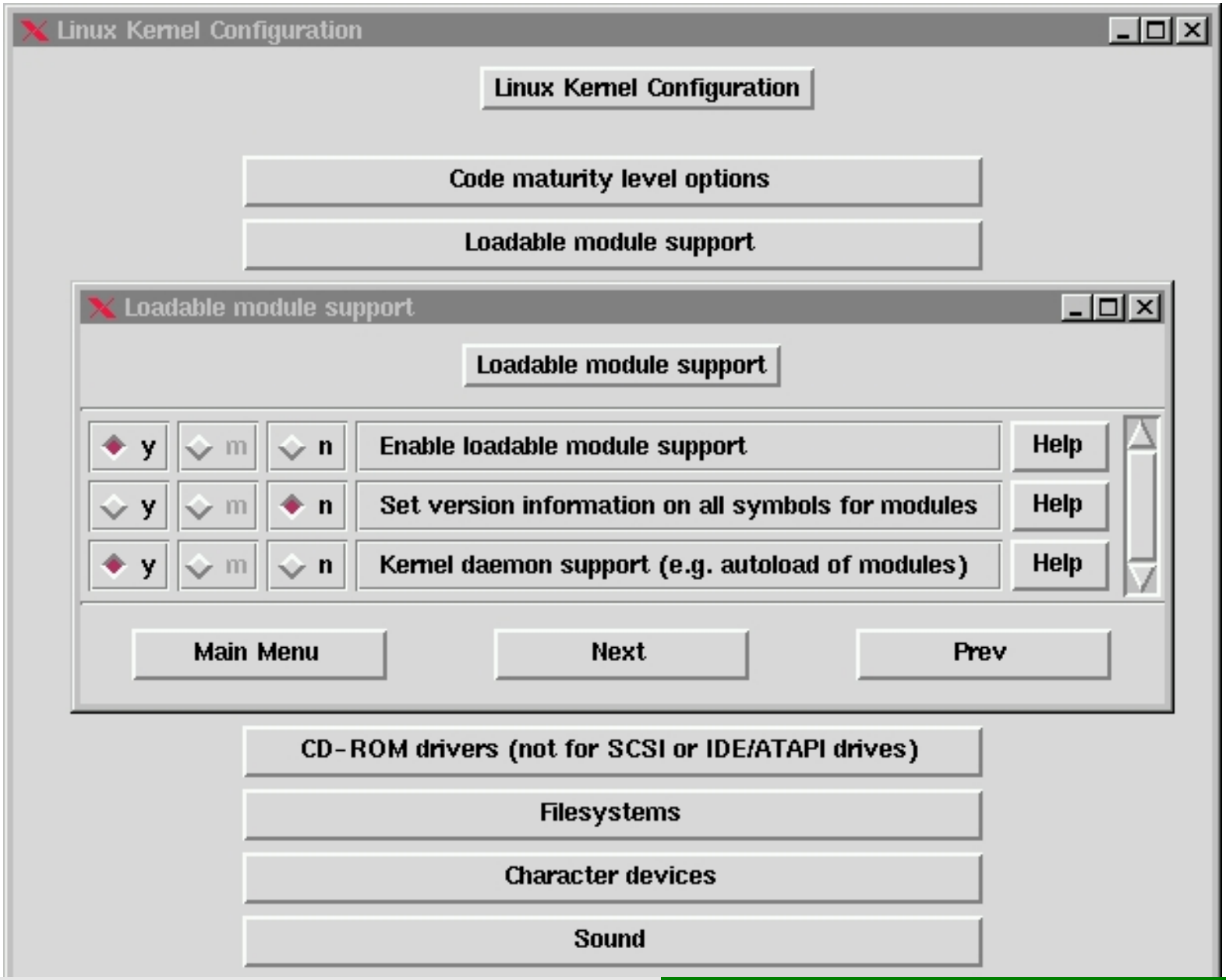
Exemple (make config)

```
# make config
...
*
* Loadable module support
*
Enable loadable module support (CONFIG_MODULES) [Y/n/?]
Set version information on all symbols for modules (CONFIG_MODVERSIONS)
Kernel module loader (CONFIG_KMOD) [Y/n/?]
*
* General setup
*
Networking support (CONFIG_NET) [Y/n/?]
PCI support (CONFIG_PCI) [Y/n/?]
...
```

Exemple (make menuconfig)



Exemple (make xconfig)



Tout est fichier

Les types de fichiers peuvent être :

- un fichier régulier
- un répertoire
- un tube nommé
- une socket
- un lien symbolique
- des fichiers spéciaux :
  - en mode bloc (essentiellement les disques)
  - en mode caractère

  Ces fichiers spéciaux se trouvent usuellement dans `/dev` et représentent un périphérique.

Exemple : `/dev/lp` représente l'imprimante.

Fichier en mode bloc

Lors'un processus acède à un périphérique en mode bloc :

- essentiellement les disques
- lire un "bloc" sur un fichier spécial
- le bloc va être chargé dans une mémoire tampon (disk buffer)
- lire la mémoire tampon
- si un autre processus veut accéder à ce même bloc sur le disque, il lira sur la mémoire tampon (bloc déjà chargé)
- l'écriture se fait dans le buffer. Lorsqu'il est plein, on reporte des blocs sur le disque

```
0 brw-r-----   2 root  operator  116,   0x00010002 12 oct  2004 ad0
0 brw-r-----   2 root  operator  116,   0x00020000 12 oct  2004 ad0s1a
```

Fichier en mode caractère

- permet de lire un nombre quelconque de caractères
- avantage ; moins de transition qu'en mode bloc
- périphériques : souris, clavier, écran
- les diques possèdent aussi une interface en mode caractère,
  ex : copie disque à disque : **dd /dev/rad0 /dec/rad1**

```
0 crw-r-----   2 root  operator  116, 0x00010002 12 oct  2004 rad0
0 crw-------   1 root  wheel      12, 128 12 oct  2004 sysmouse
0 crw-------   1 root  wheel      16,   0 12 oct  2004 lpt0
```

Pilote

le pilote de périphérique (device driver) : est un logiciel qui sait manipuler ce périphérique.
Exemple : `/dev/lp` représente l'imprimante.

```
echo "abcdef" > /dev/lp
```

Lorsque le processus écrit dans `/dev/lp` :

- les caractères sont transmis au pilote
- le pilote les envoie à l'imprimante.

Il y a un pilote par type de périphérique (disque dur, bus SCSI, ligne série, port parallèle, lecteur de bande, etc.)

Ajout de périphérique

- Pilotes de périphériques caractérisés par :
  - un type (caractère ou bloc)
  - un numéro majeur
- Le numéro majeur est utilisé comme indice dans une table interne du noyau
- Ajout d'un pilote :
  - intégration du pilote dans une table du noyau
  - modification du fichier de configuration
  - regénération d'un noyau

Fichiers de configuration

- Solaris 2 : `/usr/kernel/drv/*conf`, `/usr/kernel/drv/*`
- HP-UX 9 : `/etc/master`, `/etc/conf/dfile`
- HP-UX 10 : `/usr/conf/master.d/*`, `/stand/system`
- SunOS : `/sys/sunX/conf/NOYAU`, `/sys/sunX/conf/files*`
- OSF/1 : `/sys/conf/NOYAU`, `/sys/conf/files*`
- FreeBSD : `/sys/i386/conf/NOYAU`, `/sys/i386/conf/files*`

Intégration dans BSD (1)

- Ajout d'une entrée dans `files.`*`NOYAU`* :

```
local/pilote.o   optional périph      device-driver
```

- Placement des objets dans `/sys` :

```
mkdir /sys/local
cp pilote.o /sys/local/pilote.o
```

- Intégration dans une table : édition de conf.c
  - table des périphériques en mode caractère : `cdevsw`
  - table des périphériques en mode bloc : `bdevsw`

Intégration dans BSD (2)

- Exemple :

```
extern int drv_open(), drv_close(), drv_read();
...
struct cdevsw cdevsw[] =
{
    ...
    {    drv_open, drv_close, drv_read, nodev,
         nodev,    nodev,     nodev,    0,
         nodev,    0,         0,
    },
```

- Modification du fichier de configuration :

```
device-driver   driver
```

- Reconstruction du noyau

Fichiers spéciaux

- Création des fichiers spéciaux correspondant au(x) périphérique(s) dans le répertoire `/dev`
- `mknod fichier type majeur mineur`
- Scripts de création :
  - `/dev/MAKEDEV`
  - `/dev/MAKEDEV.local`
- Exemple :

```
cd /dev
./MAKEDEV pty
```

Ajout d'un nouveau périphérique

- configuration matérielle (exemple, brancher le joystick)
- récupérer (ou écrire) le source ou le `.o` du pilote de votre matériel. exemple : `joy.c` et `joy.h` ou directement `joy.o`
- copier dans le répertoire source des sources du noyau (de préférence au bon endroit) exemple : **cp joy.h joy.c /usr/src/sys/i386/isa/**
- ajouter à la liste des pilotes de périphériques : `/usr/src/sys/conf/files.i386` la ligne suivante :

```
i386/isa/joy.c   optional  joy   device-driver
```

- dans le fichier `/usr/src/sys/i386/conf.c`

```
#include "joy.h"

struct cdevsw [] {
}
```

L'ordre dans le tableau donnera le numéro du majeur

Ajout d'un nouveau périphérique (suite)

- générer un nouveau noyau
  - ajouter dans le fichier de configuration :

    ```
    device joy0 at isa? PORT "IO_GAME"
    ```

  - config, compiler, installer et redémarrer.
- créer le fichier spécial :

```
0 crw-------  1 root  wheel   21,   0 12 oct  2004 /dev/joy0
```

Création du fichier spécial

```
0 crw-------  1 root  wheel   21,   1      12 oct  2004 /dev/joy0
                              ---   ---
                               |     |
                          num majeur  num mineur
                          --------------------
                               32 bits
```

- numéro majeur : identifie le pilote par rapport au système
- numéro mineur : n'a de signification que pour le pilote pour savoir vers quel périphérique s'adresser
- (majeur, mineur) = device number
- sous BSD : 8 bits (256) pour le majeur, 24 bits (1677216) pour le mineur
- on utilise les bits supplémentaires du mineur pour réaliser des opérations autre que la lecture (read) ou l'écriture (write) avec **ioctl ()**

Création du fichier spécial (suite)

Créer le fichier spécial : **mknod**

- mknod nom (b ou c) majeur mineur
- exemple : mknod joy0 c 21 0

ou si c'est un périphérique déjà "connu" par les développeurs du système : script **/dev/MAKEDEV** : **./MAKEDEV joy0**

Noyau : Partie statique et Modules

Le noyau est composé d'une **partie statique** à laquelle on peut dynamiquement greffer des **modules**.

- La partie statique est utilisée lors du démarrage de votre ordinateur et sera toujours chargée en mémoire
- Les modules peuvent être chargés seulement une fois la machine démarrée et uniquement en cas de besoin.

Modules

- Module : sous-système chargé dynamiquement en mémoire
- Pas contenu de manière statique dans le noyau
- Supportés par SunOS, Solaris 2, IRIX, *BSD, Linux.
- Deux types de chargements :
  - manuel
  - à la demande

Compilation du module

Gestion des modules (1)

- Solaris 2 :
  - chargement : `modload`
  - suppression : `modunload`
  - liste : `modinfo`
- SunOS :
  - chargement : `modload`
  - suppression : `modunload`
  - liste : `modstat`
- IRIX : `ml`

Gestion des modules (2)

- *BSD :
  - chargement : `kldload`
  - suppression : `kldunload`
  - liste : `kldstat`
- Linux :
  - modules situés dans `/lib/modules/`*`version`*
  - chargement : `insmod`
  - suppression : `rmmod`
  - dépendances : `depmod`, `modprobe`
  - chargement à la demande : `kmod`

Démarrage d'un noyau Linux

- Noyau chargé par un programme externe
- Généralement LILO (LInux LOader)
- Installation d'un nouveau noyau :
  - copie de l'image (`arch/i386/boot/zImage` ou `arch/i386/boot/bzImage`)
  - configuration de `loadlin`, `lilo` ou de `grub`

LILO

- configuration de `lilo` : édition de `/etc/lilo.conf`
- exécution de `lilo` pour prendre les changements en compte

Exemple de fichier /etc/lilo.conf

```
boot=/dev/hda
map=/boot/map
install=/boot/boot.b
prompt
timeout=50
read-only
root=/dev/hda1
image=/boot/vmlinuz-2.2.14
        label=2.2.14
image=/boot/vmlinuz-2.2.13
        label=2.2.13
other=/dev/hda2
        label=freebsd
        table=/dev/hda
```

GRUB

`/boot/grub/menu.lst` lu à chaque démarrage par GRUB.
⇒ Pas de réinstallation de GRUB (contrairement à LILO)

Exemple de fichier /boot/grub/menu.lst

Attention, GRUB numérote les partitions à partir de 0 et Linux à partir de 1.

```
timeout 10     # Démarrer sur l'entrée par défaut au bout de
default 0      # Numéro de l'entrée par défaut

# Entrée 0
title GNU/Linux    # Titre qui apparaîtra au démarrage
root (hd0,1)       # Partition racine (1er disque, 2ème part
kernel /boot/vmlinuz-2.6.17-10-386 root=/dev/hda2 read-only
          # Nom de l'image du noyau et partition racine

# Entrée 1
title Windows
root (hd0,0)        # Partition racine (1er disque, 1ère par
makeactive
chainloader +1
```

Crédits

Une partie de ce cours a été inspiré par le cours de Rémy Card de l'université de Versailles-Saint-Quentin.